PENGELOMPOKAN WILAYAH BERDASARKAN KESEJAHTERAAN SOSIAL MENGGUNAKAN ALGORITME *SELF-ORGANIZING MAPS* DENGAN PERBAIKAN *MISSING VALUE K-NEAREST NEIGHBORS*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Dese Narfa Firmansyah
NIM: 155150201111153

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2019

PENGESAHAN

PENGELOMPOKAN WILAYAH BERDASARKAN KESEJAHTERAAN SOSIAL MENGGUNAKAN ALGORITME *SELF-ORGANIZING MAPS* DENGAN PERBAIKAN *MISSING VALUE K-NEAREST NEIGHBORS*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh :
Dese Narfa Firmansyah
NIM: 155150201111153

Skripsi ini telah diuji dan dinyatakan lulus pada
19 Juli 2019
Telah diperiksa dan disetujui oleh:

Pembimbing I	Pembimbing II
Sigit Adinugroho, S.Kom., M.Sc.	Bayu Rahayudi, S.T., M.T.
NIK: 201607 880701 1 001	NIP: 19740712 200604 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D.
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar referensi.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 29 Juli 2019

Dese Narfa Firmansyah

NIM: 155150201111153

PRAKATA

Puji syukur penulis panjatkan ke-hadirat Allah SWT. Karena atas limpahan rahmat dan karunia-Nya, penulis dapat menyelesaikan laporan skripsi yang berjudul "Pengelompokan Wilayah Berdasarkan Kesejahteraan Sosial Menggunakan Algoritme *Self-Organizing Maps* Dengan Perbaikan *Missing Value K-Nearest Neighbors"*. Laporan skripsi ini disusun sebagai syarat untuk memperoleh gelar sarjana pada Program Studi S1 Teknik Informatika, Jurusan Teknik Informatika, Fakultas Ilmu Komputer (FILKOM), Universitas Brawijaya.

Dalam menyusun dan menyelesaikan skripsi ini, penulis menyadari bahwa skripsi ini tidak mungkin berhasil disusun sedemikian ini tanpa bantuan dari beberapa pihak. Oleh karena itu, penulis mengucapkan terima kasih pada:

1. Sigit Adinugroho, S.Kom., M.Sc. dan Bayu Rahayudi, S.T., M.T. selaku dosen pembimbing 1 dan dosen pembimbing 2 yang telah membimbing dan memberikan arahan kepada penulis dalam menyelesaikan laporan skripsi ini,
2. Bapak Agus Wahyu Widodo, S.T., M.Sc. selaku Ketua Program Studi Teknik Informatika, Fakultas Ilmu Komputer (FILKOM) Universitas Brawijaya,
3. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D. selaku Ketua Jurusan Teknik Informatika, Fakultas Ilmu Komputer (FILKOM) Universitas Brawijaya,
4. Bapak Sunarto dan Ibu Ulfa selaku orang tua yang telah membesarkan penulis dengan kasih sayang, perhatian dan kesabarannya serta keluarga besar dari penulis yang telah memberikan dukungannya kepada penulis baik moral maupun secara materi,
5. Seluruh *civitas academica* dari Fakultas Ilmu Komputer (FILKOM), Universitas Brawijaya yang juga telah memberi banyak bantuan berupa ilmu yang bermanfaat selama penulis menempuh program studi S1 Teknik Informatika, Universitas Brawijaya.

Penulis juga menyadari bahwa laporan ini tidak pernah lepas dari yang namanya kesalahan dan kekurangan. Penulis sangat menerima segala kritik dan saran yang membangun sebagai bahan untuk meningkatkan dan memperbaiki kualitas diri. Penulis juga berharap semoga skripsi ini dapat memberikan manfaat bagi siapapun yang membacanya.

Malang, 29 Juli 2019

Penulis

desenfirman@gmail.com

ABSTRAK

Dese Narfa Firmansyah, Pengelompokan Wilayah Berdasarkan Kesejahteraan Sosial Menggunakan Algoritme *Self-Organizing Maps* Dengan Perbaikan *Missing Value K-Nearest Neighbors*

Pembimbing: Sigit Adinugroho, S.Kom., M.Sc. dan Bayu Rahayudi, S.T., M.T.

Penyandang Masalah Kesejahteraan Sosial (PMKS) merupakan kelompok sosial yang tidak dapat melaksanakan fungsi sosialnya dan hidup dibawa garis kesejahteraan masyarakat. PMKS merupakan salah satu komponen dalam penentuan kelompok sasaran program yang ada di Jawa Timur. Penelitian ini dilakukan untuk mengetahui bagaimana kelompok sasaran berdasarkan daerah yang ada di Jawa Timur dengan atribut data PMKS. Metode yang diusulkan yaitu *clustering* dengan menggunakan algoritme *Self-Organizing Maps* (SOM) dan penggunaan teknik pengisian data kosong *K-Nearest Neighbors* (KNN) untuk mengatasi permasalahan data PMKS yang beberapa diantaranya kosong. Alur kerja metode yaitu *dataset* dilakukan pengisian data kosong dengan KNN terlebih dahulu. Sebelum proses *training* menggunakan SOM dimulai, normalisasi *min-max* dilakukan pada *dataset*. Selanjutnya, proses *training* dijalankan dan hasil dari algoritme SOM merupakan keanggotaan *cluster* beserta nilai *Silhouette Coefficient*. Pengujian parameter algoritme dilakukan untuk mengetahui parameter terbaik algoritme ketika diukur menggunakan *Silhoutte Coefficient*. Pengujian mendapatkan parameter terbaik untuk algoritme SOM dengan nilai *alpha*=0,1, *eta*=0,2, jumlah *epoch*=160 dan jumlah neuron=2x2. Sedangkan untuk KNN, parameter terbaik yang didapatkan yaitu nilai K sebesar 2. K=2 memberikan kenaikan pada *Silhouette Coefficient* sebesar 3,4% dibandingkan dengan pengelompokan tanpa pengisian data kosong. Dengan menggunakan parameter tersebut, proses pengelompokan daerah atau *clustering* memberikan nilai *Silhouette Coefficient* tertinggi sebesar 0,351 yang termasuk ke dalam kategori *weak structure*. Hasil dari *clustering* tersebut terdiri dari 2 *cluster* dengan proporsi data 1:37. *Cluster* 1 hanya diisi oleh Kabupaten Jember dan *cluster* 2 diisi oleh sisa daerah yang lain. 5 atribut yang memiliki selisih tertinggi antara kedua *cluster* tersebut yaitu Lanjut Usia Terlantar, Gelandangan dan Gelandangan Psikotik, Pemulung, Pengemis dan Kelompok Minoritas.

Kata kunci: Penyandang Masalah Kesejahteraan Sosial (PMKS), clustering, pengisian data kosong, Self-Organizing Maps, K-Nearest Neighbors

ABSTRACT

Dese Narfa Firmansyah, *Region Grouping in East Java based on Person with Social Welfare Problems using Self-Organizing Maps Algorithm and K-Nearest Neighbors Missing-Value Imputation.*

Supervisors: Sigit Adinugroho, S.Kom., M.Sc. and Bayu Rahayudi, S.T., M.T.

*Persons with Social Welfare Problem (*PMKS*) are social groups that cannot do their social function because obstacle and their life are under a welfare line.* PMKS *is one of component to determine programs in East Java Government. This study aim to find out region based groups in East Java with* PMKS *data feature. The method proposed in this study are clustering with Self-Organizing Maps (SOM) algorithm and K-Nearest Neighbors (KNN) missing-value imputation in order to overcome missing-value problem in* PMKS *dataset. First, empty data in dataset is filled using KNN. Before the training process using SOM starts, min-max normalization is done on the dataset. Furthermore, the training process is carried out and the results of the SOM algorithm are cluster membership along with the Silhouette Coefficient value. Parameter testing is done to find out best value of parameter of algorithm and it's evaluated using Silhouette Coefficient. The testing gets their best parameter on SOM algorithm with learning rate value of 0.1, neighborhood coefficient value of 0.2, number of epoch is 160 and neuron size is 2x2. As For KNN, the best parameter obtained is K=2. K=2 gives increase in Silhouette Coefficient score of 3.4% compared to clustering without KNN missing-value imputation. By using these best parameters, the clustering provides highest Silhouette Coefficient 0.351 which categorized as weak structure. The result of clustering consist of 2 cluster with proportion of data 1:37. Cluster 1 is only filled by* Kab. Jember *and Cluster 2 is filled by rest of the region. The 5 attributes that have the highest difference between the two clusters are Neglected Elderly, Homeless and Psychotic Homeless, Scavengers, Beggars and Minority Groups.*

Keywords: Persons with Social Welfare (PMKS)*, clustering, missing-value imputation, Self-Organizing Maps, K-Nearest Neighbors*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE SUMBER

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Penyandang Masalah Kesejahteraan Sosial (PMKS) adalah kelompok sosial yang mencakup perseorangan, keluarga, kelompok dan/atau masyarakat yang tidak dapat melaksanakan fungsi sosialnya karena suatu hambatan, kesulitan atau gangguan (Permensos, 2012). Secara umum, PMKS adalah kelompok sosial yang hidup di bawah garis kesejahteraan masyarakat.

Gubernur Jawa Timur periode 2015-2018, Soekarwo, menyatakan bahwa pengurangan pengangguran dan kemiskinan adalah salah satu pertumbuhan ekonomi inklusif (Faiq & Parmin, 2018). Strategi yang dilakukan Provinsi Jawa Timur yaitu menerapkan program berbasis *Pro Poor*, *Pro Job*, *Pro Growth* dan *Pro Environment* untuk menanggulangi kemiskinan. Prioritas sasaran kelompok atau *cluster* yang ditargetkan untuk tahun 2018 yaitu bantuan sosial, pemberdayaan masyarakat, pemberdayaan usaha ekonomi, serta *cluster* murah pro rakyat. Sasaran kelompok ini digunakan sebagai pendukung keputusan dalam menentukan kebijakan untuk mengentas Penyandang Masalah Kesejahteraan Sosial. Selain sasaran kelompok prioritas yang ditentukan oleh pemerintah daerah Provinsi Jawa Timur, Kementerian Sosial Republik Indonesia juga membagi PMKS ke dalam beberapa kelompok sasaran. Kelompok sasaran tersebut antara lain kelompok kemiskinan, ke-terlantaran, kecacatan, keterpencilan, ketunaan dan penyimpangan perilaku, korban bencana dan kelompok KTK, eksploitasi dan diskriminasi (Departemen Sosial RI, 2012).

Salah satu cara yang dapat dilakukan untuk menentukan prioritas kelompok sasaran atau *cluster* tersebut yaitu dengan melakukan pengelompokan wilayah berdasarkan kesamaan nilai variabel atau fitur data dari PMKS. Teknik pengelompokan data atau *clustering* dapat menjadi solusi untuk memetakan persebaran data PMKS berdasarkan kabupaten/kota dengan tujuan agar kebijakan yang dibuat selanjutnya lebih tepat sasaran. Teknik pengelompokan data yang berbasis *clustering* juga diharapkan dapat dijadikan sebagai pembanding dan evaluasi terhadap sasaran kelompok yang telah ditentukan sebelumnya. Pada salah satu penelitian, algoritme SOM digunakan untuk melakukan visualisasi data dengan konsep *clustering* pada data PMKS di Daerah Istimewa Yogyakarta (Firdauz & Widodo, 2018). *Clustering* digunakan sebagai alat dalam menunjang sebuah kebijakan terkait PMKS. Penelitian tersebut juga melakukan analisis kesamaan variabel di daerah pada sebuah *cluster*. Sehingga harapannya, kebijakan yang ditentukan oleh pemerintah akan lebih tepat sasaran karena mempertimbangkan kesamaan variabel-variabel pada beberapa daerah dalam satu *cluster* tersebut.

Algoritme *Self-Organizing Map* (SOM) merupakan salah satu teknik pembelajaran mesin kategori *unsupervised learning* berbasis *neural network* yang memiliki tujuan untuk melakukan visualisasi data dari data berdimensi

tinggi ke data ke dimensi rendah. Penelitian-penelitian sebelumnya menggunakan algoritme SOM sebagai teknik untuk merepresentasikan data PMKS (yang berdimensi tinggi) ke dalam data berdimensi rendah yang berbentuk *cluster*. Penelitian tersebut menunjukkan bahwa algoritme SOM dapat digunakan untuk melakukan *clustering* (Hafiludien & Istiawan, 2018). Hasil evaluasi terbaik pada penelitian tersebut yaitu didapatkan jumlah *cluster* sebanyak 3 dengan tolak ukur berupa *Davies-Bouldin Index* (DBI).

Selain menggunakan DBI, salah satu cara untuk melakukan evaluasi *cluster* adalah menggunakan *Silhouette Coefficient*. Salah satu contoh penerapannya yaitu, *Silhouette Coefficient* digunakan untuk melakukan pengukuran atau evaluasi *clustering* menggunakan algoritme *Self-Organizing Maps* dengan studi kasus data berupa dokumen. Pada penelitian tersebut, nilai rata-rata global *Silhouette Coefficient* digunakan untuk melakukan perbandingan algoritme *clustering*. Ketiga algoritme yang dibandingkan yaitu *K-Means, Bisecting K-Means* dan *Self-Organizing Maps* (Gharib *et al.*, 2012).

Pada fakta di lapangan, data PMKS yang didapatkan memiliki banyak nilai yang kosong. Nilai kosong tersebut berupa ketiadaan nilai pada suatu daerah untuk fitur data tertentu. Data yang kosong ini dapat diisi dengan beberapa nilai, bisa menggunakan nilai 0, *mean* ataupun *median* (Gorunescu, 2011). Selain teknik tersebut, pengisian data kosong juga dapat menggunakan algoritme berbasis *machine learning*. Pada penelitian sebelumnya, metode imputasi data ini telah diterapkan pada masalah deteksi kanker payudara berbasis *machine learning* (Jerez *et al.*, 2010). Pada penelitian tersebut, teknik imputasi data memberikan hasil peningkatan akurasi yang signifikan hingga 30% pada algoritme *Missing-Data Imputation* berbasis *K-Nearest Neighbors* (KNN), *Multi-Layer Perceptron* (MLP) dan *Self-Organizing Maps*.

Dari hasil penelitian sebelumnya, penelitian kali ini akan menggunakan metode *clustering* menggunakan algoritme *Self-Organizing Maps (*SOM) dan metode Imputasi Data berbasis *K-Nearest Neighbors* (KNN). Penggunaan algoritme SOM pada penelitian ini yaitu sesuai dengan tujuan awal yaitu melakukan *clustering*. Metode imputasi data menggunakan algoritme KNN dipilih karena algoritme tersebut memberikan hasil terbaik dibandingkan dengan metode imputasi data yang lain (*Multi-Layer Perceptron* dan *Self-Organizing Maps imputation data*) sekitar 5-10% lebih tinggi. Penggunaan teknik imputasi data pada SOM diharapkan dapat memberikan peningkatan nilai *Silhouette Coefficient* pada proses *clustering*. Sehingga pada keberlanjutan penelitian ini, penelitian difokuskan untuk melakukan implementasi algoritme clustering SOM serta melakukan implementasi pengisian *missing-value* pada proses *clustering* dan melihat perbedaan hasil antara *clustering* dengan metode pengisian *missing-value* dan *clustering* tanpa metode pengisian *missing-value*.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan pada subbab sebelumnya, maka rumusan masalah yang akan menjadi fokus pada penelitian ini yaitu:

1. Bagaimana parameter terbaik algoritme SOM dalam pengelompokan data Penyandang Masalah Kesejahteraan Sosial (PMKS) berdasarkan wilayah di Provinsi Jawa Timur tahun 2016?
2. Bagaimana pengaruh imputasi data terhadap kualitas *cluster* yang terbentuk?
3. Bagaimana hasil analisis *cluster* dari penerapan SOM untuk data PMKS Provinsi Jawa Timur ketika diuji dan diukur dengan skala *Silhouette Coefficient*?

1.3 Tujuan

Pada penelitian ini, tujuan dibagi menjadi tujuan umum dan tujuan khusus. Berdasarkan rumusan masalah yang telah didefinisikan sebelumnya, tujuan-tujuan yang harus dicapai pada penelitian ini yaitu:

1. Melakukan pengujian beberapa parameter pada algoritme SOM ketika digunakan untuk pemetaan data PMKS Provinsi Jawa Timur tahun 2016.
2. Menguji dan melakukan analisis pengaruh penggunaan metode pengisian *missing-value* pada proses *clustering* dengan algoritme SOM.
3. Melakukan analisis pada *cluster* yang telah terbentuk pada algoritme SOM.

1.4 Manfaat

Manfaat yang dapat diambil dari berbagai pemangku kepentingan dari hasil penelitian yang telah dilakukan yaitu:

Bagi Masyarakat:

Masyarakat dapat mengetahui bagaimana kondisi dan karakteristik dari *dataset* PMKS pada tiap-tiap daerah dengan melihat visualisasi yang dikelompokkan dalam bentuk *cluster*. Sehingga, masyarakat dapat dengan mudah melihat sebuah data PMKS karena strukturnya telah dikelompokkan berdasarkan kesamaan parameter/variabel.

Bagi Pemerintahan:

Selain dapat mengetahui bagaimana pola persebaran dari *dataset* PMKS tiap daerahnya, hasil penelitian ini diharapkan dapat dijadikan acuan dalam menentukan kebijakan selanjutnya. Kebijakan yang dibuat akan lebih mudah karena daerah yang memiliki karakteristik sama akan dikelompokkan ke dalam satu *cluster* yang sama.

1.5 Batasan Masalah

Agar penelitian yang dilakukan memiliki ruang lingkup masalah yang jelas di awal. Batasan masalah pada penelitian ini yaitu:

1. Data yang digunakan adalah data PMKS pada Provinsi Jawa Timur pada tahun 2016. Deskripsi data secara detail dapat dilihat pada Bab metodologi penelitian sub-bab deskripsi data.
2. Pada penelitian ini, topologi algoritme *Self-Organizing Maps* yang digunakan yaitu topologi *2D-Rectangular.*
3. Fungsi ketetanggaan algoritme *Self-Organizing Maps* yang digunakan pada penelitian kali ini yaitu fungsi *Gaussian Neighbors* dengan *update learning rate linear*.

1.6 Sistematika Pembahasan

Sistematika yang digunakan dalam penulisan laporan penelitian ini adalah sebagai berikut:

a. BAB I PENDAHULUAN

Bagian ini membahas deskripsi umum pada penelitian. Bab pendahuluan mencakup latar belakang penelitian, rumusan masalah, tujuan penelitian, batasan permasalahan dan sistematika pembahasan.

b. BAB II LANDASAN KEPUSTAKAAN

Pada bab ini berisi tentang penelitian-penelitian sebelumnya dan beberapa teori yang mendukung dan berhubungan dengan penelitian ini.

c. BAB III METODOLOGI PENELITIAN

Metodologi penelitian membahas tentang metode dan prosedur yang digunakan selama penelitian.

d. BAB IV PERANCANGAN

Bagian ini berisi perancangan dari algoritme yang dibangun, perancangan pengujian serta perancangan antarmuka

e. BAB V IMPLEMENTASI

Bagian ini berisi implementasi dari perancangan yang telah dilakukan sebelumnya.

f. BAB V HASIL DAN PEMBAHASAN

Bab ini berisi pengujian metode, hasil dari pengujian beserta penjelasan secara detail dari hasil penelitian.

g. BAB VI PENUTUP

Bagian ini berisi sebuah kesimpulan yang didapat pada penelitian serta memberikan saran penelitian selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

Kajian pustaka mengacu pada beberapa penelitian yang telah dilakukan sebelumnya. Bagian ini menjelaskan bagaimana relevansi antara penggunaan algoritme *Self-Organizing Maps* (SOM) dengan *clustering* serta bagaimana metode *missing-value imputation* dapat memberikan signifikansi pada sebuah algoritme pembelajaran mesin.

Penelitian sebelumnya menunjukkan bahwa Algoritme SOM dapat digunakan sebagai analisis pengelompokan atau *clustering* dengan objek penelitian yaitu data penerima beasiswa (Rahmawati *et al.*, 2015). Penelitian tersebut menggunakan nilai *Davies-Bouldin Index* (DBI) sebagai tolak ukur untuk setiap jumlah *cluster* yang akan diuji, yaitu 2, 3, 4, 5. Selain itu juga, nilai *alpha* yang digunakan pada penelitian tersebut yaitu 0,6. Pada penelitian tersebut, proses *clustering* mendapatkan bentuk *cluster* terbaik yaitu sebanyak 4 dengan skor DBI sebesar 0.098 atau lebih kecil dibanding jumlah *cluster* selanjutnya yang memiliki nilai DBI sebesar 0.34. Pengambilan jumlah *cluster* terbaik didasarkan pada jumlah *cluster* yang memiliki nilai DBI paling kecil. Kekurangan pada penelitian tersebut yaitu tidak dilakukannya pengujian nilai parameter-parameter pada algoritme SOM.

Kemudian pada penelitian sebelumnya, algoritme SOM juga digunakan untuk clustering Data PMKS Provinsi Daerah Istimewa Yogyakarta (DIY) tahun 2016 (Firdauz & Widodo, 2018). Algoritme SOM digunakan sebagai pendukung dalam menentukan kelompok-kelompok daerah yang akan ditentukan kebijakannya. Penelitian ini menggunakan topografi SOM 2D-rectangular dan melakukan analisis dan pembagian *cluster* dengan menggunakan *Fan Diagram*. *Fan* diagram disini menunjukkan kecenderungan sebuah daerah masuk ke dalam *cluster* tertentu pada representasi jaringan SOM 2D-rectangular. Pada penelitian tersebut didapatkan hasil jumlah *cluster* sebanyak 5 dengan metode pendekatan *Within Cluster Sum of Squares*. Kekurangan dari penelitian tersebut yaitu tidak melakukan proses interpretasi dan analisis dari *cluster* yang terbentuk.

Penelitian selanjutnya yaitu perbandingan antara beberapa teknik clustering (Dehuri *et al.*, 2006). Algoritme SOM dipilih untuk proses clustering karena tidak ada batasan berupa jumlah *cluster* yang dimasukkan layaknya algoritme *K-Means* maupun batasan berupa koefisien kerapatan anggota layaknya algoritme *clustering density-based.* Pada penelitian tersebut juga ditunjukkan bahwa algoritme *density-based clustering* memiliki *percent accuracy* yang lebih tinggi dibanding dengan SOM. Namun, algoritme *density-based* tersebut memerlukan jumlah iterasi yang banyak untuk mencapai ke kondisi *percent accuracy* tertinggi.

Pada algoritme SOM yang akan digunakan pada penelitian ini, pemilihan jenis fungsi update nilai learning rate dan jenis fungsi ketetanggaan juga dapat mempengaruhi kualitas jaringan SOM yang di-*training* (Natita *et al.*, 2016). Penelitian tersebut, menunjukkan perbandingan antara ketiga jenis proses *update learning rate* yaitu (1) *linear*, (2) *inverse* dan (3) *power series*. Sedangkan,

fungsi ketetanggaan yang dibandingkan pada penelitian tersebut yaitu (1) *Bubble* dan (2) *Gaussian*. Hasil penelitian menunjukkan bahwa kombinasi fungsi ketetanggaan *Gaussian* dan *update learning rate linear* memberikan nilai *Quantization Error* (QE) yang kecil dan stabil pada setiap *epoch*-nya.

Teknik imputasi data merupakan sebuah teknik untuk mengisi nilai dari sebuah data dengan *missing-value*. Teknik imputasi dapat digunakan untuk meningkatkan akurasi dari masalah deteksi kanker payudara (Jerez *et al.*, 2010). Penelitian tersebut membandingkan 5 teknik imputasi data yaitu *Mean*, *Hot-Deck*, MLP, KNN dan SOM. Pada penelitian tersebut didapatkan bahwa teknik imputasi data menggunakan KNN meningkatkan akurasi deteksi sebesar 30%. Nilai ini jauh signifikan dibandingkan dengan menggunakan *Mean* dan *Hot-Deck* imputation. Namun, nilai tersebut juga tidak jauh berbeda secara signifikan dengan metode imputasi data MLP dan SOM yang memiliki nilai peningkatan sekitar 1-5 % di bawah metode imputasi data yang menggunakan KNN.

2.1 Penyandang Masalah Kebutuhan Sosial

Penyandang Masalah Kebutuhan Sosial (PMKS) adalah kelompok perseorangan, keluarga, dan/atau masyarakat yang tidak dapat melaksanakan fungsi sosialnya karena suatu hambatan, kesulitan atau gangguan. Sehingga mereka hidup dengan tidak dapat memenuhi kebutuhan jasmani, rohani maupun sosial secara memadai. Data PMKS memiliki kriteria yaitu kelompok dengan masalah sosial berupa kemiskinan, tuna sosial, kecacatan, keterlantaran, keterpencilan serta korban tindak kekerasan dan diskriminasi (Permensos, 2012).

2.1.1 Jenis, Definisi dan Kriteria dari Data PMKS

Kriteria dari data PMKS dapat dikelompokkan menjadi beberapa kategori sebagai berikut:

1. Anak balita terlantar berusia 5 tahun yang kriteria utamanya yaitu tanpa asuhan yang layak, berasal dari keluarga yang dikategorikan miskin, kehilangan hak asuh dari orang tua ataupun keluarga. Anak balita dengan gizi buruk atau kurang juga masuk ke dalam kategori ini.
2. Anak terlantar merupakan anak yang berusia berusia 6 (enam) tahun sampai dengan 18 (delapan belas) tahun. Memiliki kriteria yang hampir sama dengan anak balita terlantar yaitu berasal dari keluarga kurang mampu dan kebutuhan dasarnya tidak terpenuhi
3. Anak yang berhadapan dengan hukum dengan kriteria utamanya yaitu anak yang disangka, didakwa dan dijatuhi pidana karena melakukan tindak pidana atau menjadi korban tindak pidana itu sendiri. Kategori ini meliputi anak yang berusia 6 (enam) tahun sampai dengan 18 (delapan belas) tahun.
4. Anak jalanan yaitu anak yang bekerja di jalanan untuk memenuhi kebutuhan sehari-harinya. Kriteria utamanya yaitu menghabiskan

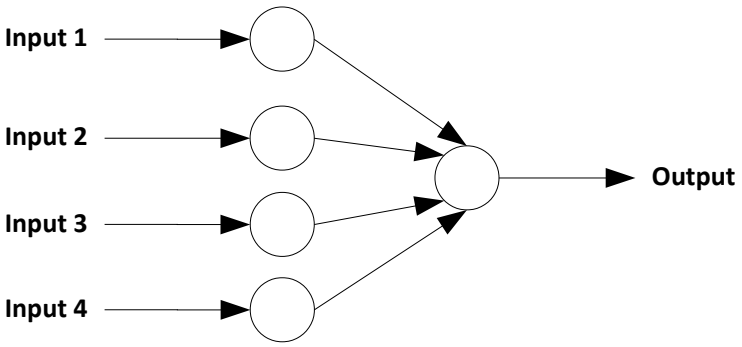
sebagian waktunya berkeliaran di jalanan dengan motif mencari nafkah maupun motif lain.

5. Anak dengan kedisabilitasan merupakan anak sebelum usia 18 (delapan belas) tahun yang mempunyai kelainan fisik dan mental.
6. Anak korban tindak kekerasan merupakan anak yang terancam baik fisik maupun non fisik karena tindak kekerasan dari lingkungan sosial terdekatnya.
7. Anak yang memerlukan perlindungan khusus
8. Orang lanjut usia terlantar
9. Penyandang kedisabilitasan
10. Tuna Susila merupakan seseorang melakukan hubungan seksual dengan sesama atau lawan jenis di luar pernikahan secara berulang-ulang dan/atau bergantian dengan tujuan mendapatkan imbalan
11. Gelandangan merupakan orang yang hidup dalam keadaan tidak layak
12. Pengemis merupakan orang yang mendapat penghasilan dengan meminta-minta
13. Pemulung merupakan orang yang melakukan pekerjaan dengan memungut barang-barang bekas
14. Kelompok minoritas merupakan kelompok yang mengalami tindakan diskriminasi yang terjadi di masyarakat.
15. Bekas Warga Binaan Lembaga Pemasyarakatan (BWBLP) merupakan orang yang telah selesai menjalani masa pidananya di Lembaga Pemasyarakatan (LP)
16. Orang dengan HIV/AIDS
17. Korban penyalahgunaan NAPZA
18. Korban trafficking merupakan seseorang yang mengalami penderitaan psikis, mental, fisik yang diakibatkan oleh perdagangan orang
19. Korban tindak kekerasan
20. Pekerja Migran Bermasalah Sosial
21. Korban Bencana Alam
22. Korban Bencana Sosial merupakan korban yang diakibatkan oleh peristiwa atau konflik sosial yang terjadi di masyarakat.
23. Perempuan rawan sosial ekonomi
24. Fakir Miskin
25. Keluarga dengan masalah sosial psikologis merupakan kelompok yang fungsi dan tugas-tugas keluarganya tidak terpenuhi

26. Komunitas adat terkecil merupakan kelompok sosial budaya yang belum menerima jaringan dan pelayanan sosial ekonomi maupun politik.

2.2 Jaringan Saraf Tiruan

Artificial Neural Network atau Jaringan Saraf Tiruan merupakan model pembelajaran mesin yang meniru konsep dari cara kerja saraf secara biologi. Sistem jaringan saraf tiruan terdiri dari kumpulan neuron atau *node* yang berhubungan dengan neuron yang lain (Fausett, 1993). Secara umum, terdapat tiga tipe neuron, yaitu *input*, *hidden* dan *output*. Neuron *input* merupakan lapisan pertama pada jaringan SOM. Neuron *input* umumnya menerima masukan parameter yang berupa variabel atau atribut dari *dataset*. Neuron *hidden* merupakan neuron yang berada pada lapisan tengah. Lapisan ini merupakan neuron yang menerima masukan dari neuron *input* atau neuron *hidden* pada lapisan sebelumnya. Lapisan ini akan melakukan kalkulasi bobot berdasarkan lapisan sebelumnya untuk dijadikan masukan pada lapisan selanjutnya. Selanjutnya neuron *output* memberikan keluaran dari algoritme. Keluaran pada umumnya yaitu merepresentasikan nilai sebuah atribut yang akan diprediksi. Sebuah jaringan saraf tiruan sederhana yang terdiri dari *input* dan *output* dapat dilihat pada Gambar 2.1.



Gambar 2.1 Arsitektur Jaringan Saraf Tiruan Sederhana

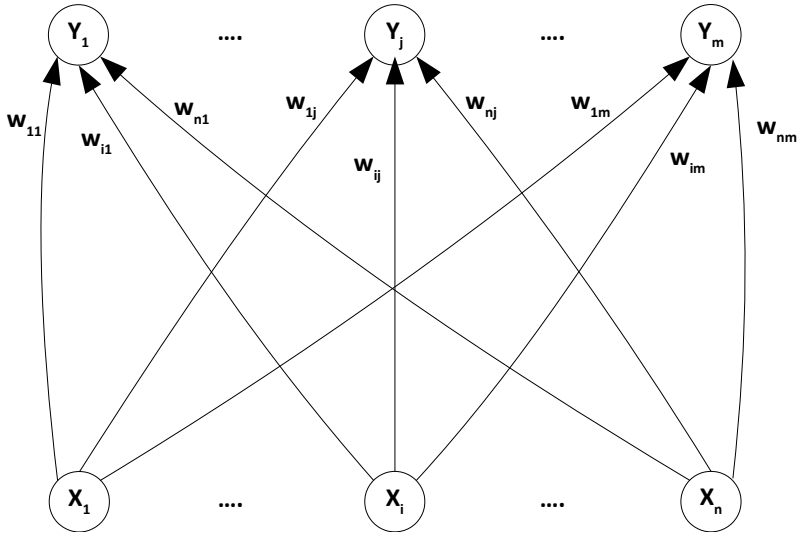
Sumber: Fausett (1993)

2.3 Algoritme *Self-Organizing Maps*

Algoritme *Self-Organizing Maps* (SOM) merupakan jenis dari algoritme berbasis jaringan saraf tiruan dengan proses *training* secara *unsupervised* dan *competitive* (Kohonen, 1990). Bersifat *unsupervised* karena algoritme SOM tidak membutuhkan data acuan sebagai proses pembelajarannya dan bersifat *competitive* karena setiap neuron akan berkompetisi dengan neuron lain dalam satu jaringan SOM untuk melakukan perubahan bobot jika neuron tersebut memiliki jarak terdekat dengan data *input*. Selain itu juga, algoritme SOM dapat bersifat korporatif jika *update* bobot juga dilakukan pada neuron tetangga dari neuron pemenang (Natita *et al.*, 2016). Salah satu contoh aplikasi dari algoritme SOM yaitu proses *clustering* atau pengelompokan data (Hagan *et al.*, 2014).

2.3.1 Struktur dan Topologi Algoritme SOM

Arsitektur dari jaringan SOM terdiri dari dua lapisan (*layer*). Lapisan tersebut yaitu lapisan input dan output. Setiap neuron input berhubungan dengan setiap neuron output yang ada pada jaringan SOM. Secara umum struktur dari jaringan SOM dapat dilihat pada Gambar 2.2.



Gambar 2.2 Arsitektur Algoritme SOM

Sumber: Fausett (1993)

Sedangkan pada topologinya, jaringan SOM dibagi menjadi 3 jenis topologi, yaitu *linear array*, *rectangular* dan *hexagonal grid* (Fausett, 1993). Pembagian topologi ini berdasarkan jenis ketetanggaan antara neuron-neuron pada jaringan SOM.

1. Topologi *linear array* memodelkan jaringan SOM dengan neuron-neuron yang disusun urut secara linear. Arsitektur *linear array* dapat dilihat pada Gambar 2.3. Pada gambar tersebut, neuron pemenang (yang juga dapat disebut sebagai *cluster* pemenang) memiliki dua unit neuron tetangga dengan jarak sebesar R=1 dan R=2.

   * * * { * (* [#] *) * } * * *

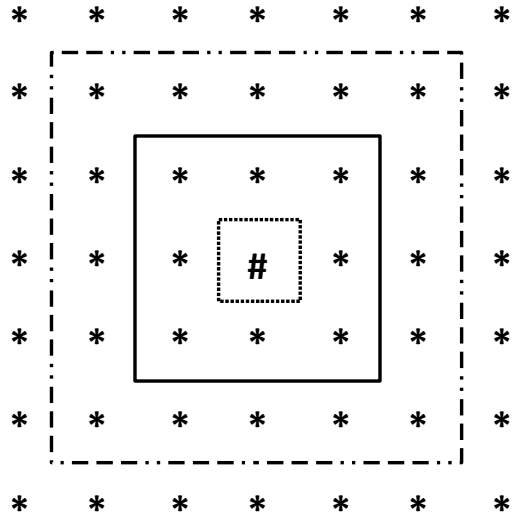
   Keterangan: [ ]: R = 0 ; ( ): R = 1; { }: R = 2

   Gambar 2.3 Topologi *Linear Array*

   Sumber: Fausett (1993)

2. Topologi *rectangular grid* memodelkan jaringan SOM ke dalam struktur neuron 2 dimensi. Pada tipe jaringan *rectangular grid,* neuron tetangga merupakan neuron-neuron yang berada di sekitar neuron pemenang dengan radius R yang berbentuk persegi. Pada Gambar 2.4, neuron yang berada di tengah (ditunjukkan oleh simbol #) memiliki 8 neuron tetangga

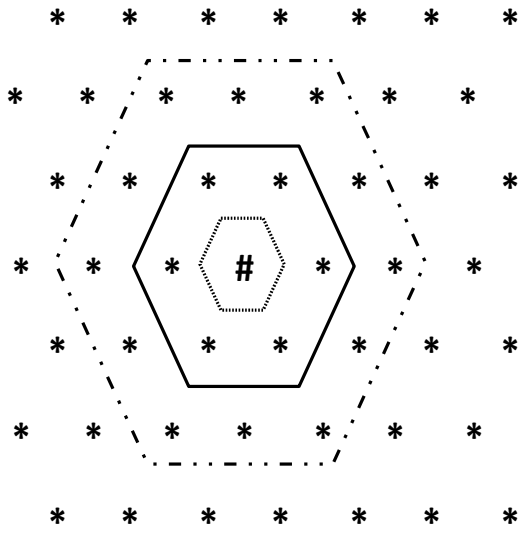
untuk R yang bernilai 1 dan memiliki 16 neuron tetangga untuk R yang bernilai 2.



Gambar 2.4 Topologi *Rectangular Grid*

Sumber: Fausett (1993)

3. Topologi *hexagonal grid* memiliki ciri yang hampir sama dengan topologi *rectangular grid.* Pada topologi *hexagonal grid,* neuron tetangga berada pada radius R dengan bentuk *hexagonal*. Pada Gambar 2.5, neuron yang berada di tengah (ditunjukkan oleh simbol #) memiliki 6 neuron tetangga untuk R yang bernilai 1 dan memiliki 12 neuron tetangga untuk R yang bernilai 2.



Gambar 2.5 Topologi *Hexagonal Grid*

Sumber: Fausett (1993)

2.3.2 Jenis Fungsi Ketetanggaan Algoritme SOM

Dalam Algoritme SOM, terdapat dua jenis fungsi dalam melakukan *update* bobot untuk neuron-neuron tetangga atau neuron yang berada di sekitar neuron pemenang. Dua fungsi tersebut yaitu fungsi *bubble neighborhood* dan *gaussian neighborhood* (Natita *et al.*, 2016).

1. Fungsi *bubble neighborhood* melakukan *update* bobot pada neuron tetangga dengan sebuah nilai yang konstan. Nilai konstan ini merupakan nilai dengan radius R yang ditentukan oleh pengguna. Fungsi tersebut dapat dilihat pada Persamaan 2.1.

$$h_{ij}^{c}=\begin{cases}\alpha(t) & ,\ (i,j)\in N_c \\ 0 & ,\ (i,j)\notin N_c\end{cases} \tag{2.1}$$

Keterangan:

h_{ij}^{c} : fungsi ketetanggan *bubble neighorhood.*

$\alpha(t)$: nilai *learning rate* atau nilai konstanta *alpha* pada algoritme SOM untuk interval waktu ke-t.

$(i,\ j)$: indeks baris ke-*i* kolom ke-*j* dari neuron pada algoritme SOM.

N_c : merupakan himpunan neuron yang berada di sekitar atau menjadi tetangga dari neuron pemenang *c*.

2. Fungsi *gaussian neighborhood* melakukan *update* bobot pada neuron tetangga dengan kriteria nilai perubahan bobot yang semakin kecil jika sebuah neuron tetangga berada jauh dengan neuron pemenang. Fungsi tersebut dapat dilihat pada Persamaan 2.2.

$$h_{ij}^{c}=\alpha(t).e^{\left(\frac{-\left\|r_{ij}-r_c\right\|^2}{2\eta_{ij}^{c}(t)^2}\right)} \tag{2.2}$$

Keterangan:

h_{ij}^{c} : fungsi ketetanggan *gaussian neighorhood.*

$\alpha(t)$: nilai *learning rate* atau nilai konstanta *alpha* pada algoritme SOM untuk interval waktu ke-*t*.

r_{ij} : letak dari neuron baris ke-*i* kolom ke-*j*.

r_c : letak koordinat dari neuron pemenang *c*.

$\eta_{ij}^{c}(t)$: nilai konstanta *eta* yang menentukan seberapa besar proses *update* bobot berpengaruh setiap tetangga yang berlangsung untuk interval waktu ke-t.

Ketika menggunakan fungsi ketetanggaan *gaussian neighborhood,* terdapat sebuah konstanta *eta* atau yang dilambangkan dengan η. Nilai dapat bersifat konstan atau berubah-ubah secara linear mengikuti interval waktu (Natita *et al.*, 2016). Perubahan nilai konstanta *eta* yang bersifat linear terhadap waktu dapat dilihat pada Persamaan 2.3.

$$\eta_{ij}^{c}(t) = \eta(0).e^{(\frac{-t}{T})} \quad (2.3)$$

Keterangan:

$\eta_{ij}^{c}(t)$: nilai *eta* untuk neuron tetangga baris ke-*i* kolom ke-*j* dari neuron pemenang *c* pada interval waktu ke-*t*.

$\eta(0)$: nilai *eta* awal yang diujikan.

t : interval waktu ke-*t*.

T : waktu atau *epoch* maksimal dari algoritme SOM.

2.3.3 Algoritme *Self-Organizing Maps* (SOM) untuk *Clustering*

Dalam penggunaan *Self-Organizing Maps* untuk permasalahan *clustering*, neuron pada algoritme SOM dapat direpresentasikan sebagai sebuah *cluster* (Hagan *et al.*, 2014). Adapun alur kerja dari penggunaan SOM untuk *clustering* adalah sebagai berikut (Fausett, 1993; Rahmawati *et al.*, 2015):

1. Melakukan inisialisasi jaringan SOM. Hal yang perlu diinisialisasi berupa, nilai konstanta *alpha* dan *beta*, jumlah *neuron* dan jumlah maksimal *epoch*/iterasi pelatihan.
2. Melakukan inisalisasi bobot jaringan w_{ij} secara random
3. Selama jumlah *epoch*/iterasi maksimal belum tercapai, maka melakukan langkah 3-7.
4. Untuk setiap data masukan, melakukan langkah 4-6.
5. Untuk setiap neuron j yang terpilih, hitung jarak neuron terhadap masing-masing variabel pada data masukan. Proses perhitungan dapat dilihat pada Persamaan 2.4.

$$D(j) = \sum_{i} (w_{ij} - x_i)^2 \quad (2.4)$$

Keterangan:

$D(j)$: jarak Euclidian antara masing-masing neuron dengan data masukan.

w_{ij} : bobot jaringan SOM untuk neuron ke-*i* parameter atau variabel ke-*j*.

x_i : data *input* untuk parameter ke-*i*.

6. Mencari *index* dari himpunan neuron j yang memiliki jarak terkecil. Pada Persamaan 2.5, c merupakan indeks dari neuron yang memiliki jarak terkecil dengan data *input*.

$$c = \underset{j}{argmin} D(j) \quad (2.5)$$

Keterangan:

c : neuron pemenang atau neuron yang memiliki jarak terdekat

dengan *input*.

$D(j)$: data jarak Euclidian pada masing-masing neuron j.

7. Setelah neuron pemenang sudah ditentukan, Untuk neuron c yang menjadi neuron pemenang dan semua neuron yang memiliki hubungan ketetanggaan dengan neuron j, maka melakukan perubahan bobot pada neuron tersebut. Nilai *learning rate* atau konstanta *alpha* yang digunakan masuk ke dalam perhitungan fungsi ketetanggaan h_{ij}^{c} (Natita *et al.*, 2016). Proses *update* bobot dapat dilihat pada Persamaan 2.6.

$$w_{ij}(new) = w_{ij}(old) + h_{ij}^{c}\left[x_i - w_{ij}(old)\right] \quad (2.6)$$

Keterangan:

$w_{ij}(new)$: nilai bobot jaringan SOM yang baru untuk neuron ke-*i* dan parameter atau variabel ke-*j*.

$w_{ij}(old)$: nilai bobot jaringan SOM yang lama untuk neuron ke-*i* dan parameter atau variabel ke-*j*.

x_i : data *input* untuk parameter ke-*i*.

h_{ij}^{c} : hasil perhitungan fungsi ketetanggaan yang telah ditentukan sebelumnya.

c : indeks dari neuron pemenang.

8. Melakukan *update* parameter penurunan nilai *learning rate* dan konstanta yang lain. Terdapat 3 jenis fungsi penurunan nilai *learning rate* yaitu (1) linear, (2) *inverse* dan (3) *power series* yang masing-masing dapat dilihat pada Persamaan 2.7, 2.8, dan 2.9 (Natita et al., 2016).

$$\alpha(t) = \alpha(0).\frac{1}{t} \quad (2.7)$$

$$\alpha(t,T) = \alpha(0).\left(1 - \frac{t}{T}\right) \quad (2.8)$$

$$\alpha(t,T) = \alpha(0).e^{\frac{t}{T}} \quad (2.9)$$

Keterangan:

$\alpha(t)$: parameter *update alpha* hanya bergantung kepada jumlah *epoch* pada selang waktu ke-*t*

$\alpha(0)$: nilai *alpha* awal

$\alpha(t,T)$: parameter *update alpha* bergantung pada jumlah *epoch* pada selang waktu ke-t dan jumlah *epoch* maksimal

t : jumlah *epoch* pada selang waktu ke-*t*

T : jumlah maksimal *epoch* pada proses *training*

2.4 *Missing Data Imputation*

Dalam metode pembelajaran data berbasis *machine learning*, sering kali terdapat beberapa nilai pada atribut data yang tidak ada. Hal ini juga dapat mempengaruhi nilai akurasi dari proses *training* yang dilakukan. Ada berbagai macam cara untuk menangani data yang tidak ada ini, salah satunya yaitu menggunakan teknik *missing data imputation* atau teknik imputasi data. Pada teknik imputasi data, daripada membuang atau membiarkan nilai yang tidak ada tersebut, nilai yang tidak ada akan di-estimasi menggunakan sebuah metode prediksi berbasis statistik ataupun *machine learning* (Gelman & Hill, 2007; Jerez *et al.*, 2010).

2.4.1 *Missing Data Imputation* menggunakan *K-Nearest Neighbors*

Metode *Missing Data Imputation* menggunakan *K-Nearest Neighbors* (KNN) yaitu mengisi nilai yang kosong berdasarkan nilai yang tidak kosong pada atribut atau variabel lain pada *dataset* (Jerez *et al.*, 2010). Metode ini memilih beberapa nilai sebanyak K pada atribut lain yang memiliki jarak terendah dengan atribut dengan nilai yang kosong tersebut. Selanjutnya, penentuan nilai yang kosong tersebut bergantung pada jenis data. Jika data adalah data diskrit, maka nilai ditentukan berdasarkan nilai yang sering muncul pada K yang dipilih. Sedangkan untuk data numerik, nilai yang digunakan yaitu rata-rata dari nilai K yang dipilih tersebut. Adapun alur kerja dari pengisian data kosong dengan menggunakan algoritme KNN adalah sebagai berikut:

1. Diberikan *vector* data *input* $x=[x_1, x_2, \dots, x_n]$
2. Menentukan jarak masing-masing *input vector* yang ada pada *dataset*. Penentuan tersebut dapat dilihat pada Persamaan 2.10.

$$d(x_a, x_b) = \sqrt{\sum_{j=0}^{n} d_j(x_{aj}, x_{bj})^2} \tag{2.10}$$

Dalam hal ini juga terdapat beberapa aturan jarak yang digunakan. Aturan tersebut dapat dilihat pada Persamaan 2.11. Nilai yang dipakai pada *denominator* atau pembagi dari persamaan tersebut yaitu nilai maksimal dan minimal pada atribut data ke-*j*. Sedangkan jika salah satu atau kedua data pada atribut ke-*j* tersebut juga kosong, maka jarak antara data tersebut bernilai 1 (Jerez *et al.*, 2010).

$$d_N(x_{aj}, x_{bj}) = \begin{cases} 1 & jika\ x_{aj}\ atau\ x_{bj}\ kosong, \\ \dfrac{|x_{aj} - x_{bj}|}{max(x_j) - min(x_j)} & jika\ x_{aj}\ dan\ x_{bj}\ tidak\ kosong \end{cases} \tag{2.11}$$

Keterangan:

x_a dan x_b : *input* data ke-*a* dan ke-*b*.

x_{aj} dan x_{bj} : *input* data ke-*a* dan ke-*b* pada atribut data ke-*j*.

j : atribut data ke-*j*.

$d_N(x_{aj}, x_{bj})$: jarak data untuk data berjenis kuantitatif.

x_a : nilai terbesar pada data untuk atribut ke-*j*.

$min(x_j)$: nilai terkecil pada data untuk atribut ke-*j*.

3. Mengurutkan data jarak antara data atribut x yang tidak ada dengan *input* data yang lain dan mengambil K jarak terdekat. Pada Persamaan 2.12, merupakan.

$$\upsilon = \{v_k\}_{k=1}^{K} \quad (2.12)$$

Keterangan:

$\{v_k\}_{k=1}^{K}$: himpunan data jarak terdekat antara data yang tidak ada dengan *input* data yang lain.

K : jumlah data jarak terdekat yang akan digunakan untuk pengisian nilai.

4. Nilai yang digunakan untuk mengisi data kosong yaitu menggunakan nilai rata-rata dari keanggotaan himpunan υ seperti pada Persamaan 2.13 (himpunan dengan K jarak terdekat). x_j merupakan data yang kosong pada atribut ke-*j*. v_{kj} merupakan tetangga K terdekat pada atribut ke-*j*.

$$\widetilde{x_j} = \frac{1}{k}\sum_{k=1}^{K} v_{kj} \quad (2.13)$$

Keterangan:

$\widetilde{x_j}$: data yang tidak ada atau *missing value* pada atribut ke-*j* yang akan diisi.

v_{kj} : data ke-*k* pada atribut ke-*j* yang digunakan untuk pengisian data kosong.

2.5 Normalisasi *Min-Max*

Proses normalisasi data merupakan proses mengubah atau melakukan transformasi data menjadi sebuah data dengan nilai *range* tertentu. Pada normalisasi *min-max,* nilai pada *dataset* dilakukan transformasi dengan ketentuan nilai tersebut berada di antara *range* maksimal dan minimal dari atribut pada *dataset* tersebut. Normalisasi *min-max* dipilih untuk menyamakan antara data yang digunakan dengan bobot jaringan SOM yang nilainya antara 0 hingga 1. Rumus dari normalisasi data dapat dilihat pada Persamaan 2.14.

$$\acute{v}(i) = \frac{v(i) - min(v(i))}{max(v(i)) - min(v(i))} \quad (2.14)$$

Keterangan:

$\acute{v}(i)$: data yang telah dinormalisasi menggunakan normalisasi *min-max.*

$v(i)$: data pada atribut ke-*i*.

$min(v(i))$: nilai terkecil pada data atribut ke-*i*.

$max(v(i))$: nilai terbesar pada data atribut ke-*i*.

2.6 *Silhouette Coefficient*

Silhouette Coefficient merupakan salah satu metode pengukuran yang digunakan untuk mengevaluasi dan memvisualisasikan kualitas dari sebuah *cluster* yang terbentuk (Rousseeuw, 1987). Setiap objek data yang terdapat pada *cluster* memiliki nilai *Silhouette Coefficient* yang diukur berdasarkan tingkat kedekatan (*cohesion)* dengan objek lain dalam satu *cluster* dan juga tingkat pemisahan (*separation*) dengan masing-masing objek di *cluster* yang lain. Tahapan perhitungan *Silhouette Coefficient* adalah sebagai berikut (Handoyo *et al.*, 2014; Hudin *et al.*, 2018):

1. Menghitung $a(i)$ yang merupakan jarak rata-rata dari sebuah objek data ke-*i* dengan dengan objek data lain yang masih satu *cluster* dengan objek ke-*i*. Perhitungan tersebut dapat dilihat pada Persamaan 2.15

$$a(i)=\frac{1}{|A|-1}\sum_{j\in A, j\neq i} d(i,j) \quad (2.15)$$

Keterangan:

$a(i)$: jarak rata-rata sebuah objek data dengan masing-masing anggota cluster A.

$d(i,j)$: jarak data ke-*i* dengan data ke-*j*.

$j\in A, j\neq i$: data ke-*j* merupakan anggota dari cluster A, dimana j adalah objek data lain yang bukan data ke-*i*.

$|A|$: banyaknya anggota cluster A.

2. Menghitung $d(i,C)$ yang merupakan jarak rata-rata objek data ke-i dengan *cluster* yang lain. Perhitungan tersebut dapat dilihat pada Persamaan 2.16.

$$d(i,C)=\frac{1}{|C|}\sum_{j\in C} d(i,j) \quad (2.16)$$

Keterangan:

$d(i,C)$: jarak data ke-*i* dengan *cluster* ke C.

$d(i,j)$: jarak data ke-*i* dengan data ke-*j*.

$j\in C$: data ke-*j* merupakan anggota dari *cluster* C

$|C|$: banyaknya anggota *cluster* C.

3. Menghitung $b(i)$ yang merupakan jarak terkecil dari masing-masing $d(i,C)$. Perhitungan tersebut dapat dilihat pada Persamaan 2.17

$$b(i)=\min_{j \in C} d(i,C) \quad (2.17)$$

Keterangan:

$b(i)$: jarak terkecil dari masing-masing $d(i,C)$ yang ada.

$d(i,C)$: jarak data ke- i dengan cluster ke C .

4. Nilai *Silhouette Coefficient* dari objek data ke-*i* dapat dihitung seperti pada Persamaan 2.18

$$s(i)=\frac{b(i)-a(i)}{max\{a(i),b(i)\}} \quad (2.18)$$

Keterangan:

$s(i)$: *Silhouette Coefficient* objek data ke- i

$b(i)$: jarak objek data ke-*i* dengan *cluster* lain yang terkecil.

$a(i)$: jarak rata-rata sebuah objek data ke-*i* dengan masing-masing anggota *cluster*.

$max\{a(i),b(i)\}$: nilai terbesar antara $a(i)$ dan $b(i)$.

Ketika sebuah *cluster* memiliki satu anggota, nilai $a(i)$ tidak dapat didefinisikan. Sehingga, *Silhouette Coefficient* dari sebuah objek data ke-*i* dapat bernilai 0 (Rousseeuw, 1987). Dari Persamaan 2.18, *Silhouette Coefficient* memiliki rentang yang dapat dilihat pada Persamaan 2.19.

$$-1 \leq s(i) \leq 1 \quad (2.19)$$

Nilai *Silhouette Coefficient* memiliki rentang antara -1 hingga +1. Objek yang memiliki nilai *Silhouette Coefficient* mendekati +1 memiliki arti bahwa objek tersebut telah dikelompokkan dengan baik. Nilai *Silhouette Coefficient* mendekati -1 menunjukkan bahwa objek tersebut masuk ke dalam *cluster* yang tidak tepat atau bisa jadi objek tersebut memiliki nilai *Silhouette Coefficient* yang bagus ketika dimasukkan ke *cluster* lain. Nilai 0 menunjukkan bahwa objek data tersebut memiliki struktur yang tidak bagus pada sebuah *cluster,* namun bukan berarti bahwa objek data tersebut masuk ke dalam *cluster* yang salah. Ilustrasi perhitungan *Silhouette Coefficient* dalam memperhitungkan jarak *inter-cluster* dan *intra-cluster* dapat dilihat pada Gambar 2.6.

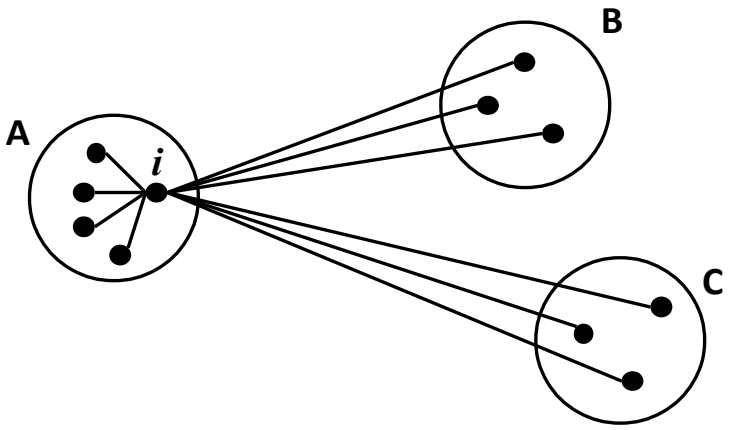
Silhouette Coefficient juga memiliki skala kualitas yang dapat dijadikan acuan dalam melakukan apakah proses clustering dikatakan berhasil atau tidak. Pembagian nilai tersebut antara lain sebagai berikut (Kaufmann & Rousseeuw, 1990; Pradnyana & Permana, 2018):

0.7 < *Silhouette Coefficient* <= 1 *: strong structure*

0.5 < *Silhouette Coefficient* <= 0.7 *: medium structure*

0.25 < *Silhouette Coefficient* <= 0.5 *: weak structure*

Silhouette Coefficient <= 0.25 *: no structure*



Gambar 2.6 Ilustrasi perhitungan pada *Silhouette Coefficient*

Sumber: Rousseeuw (1987)

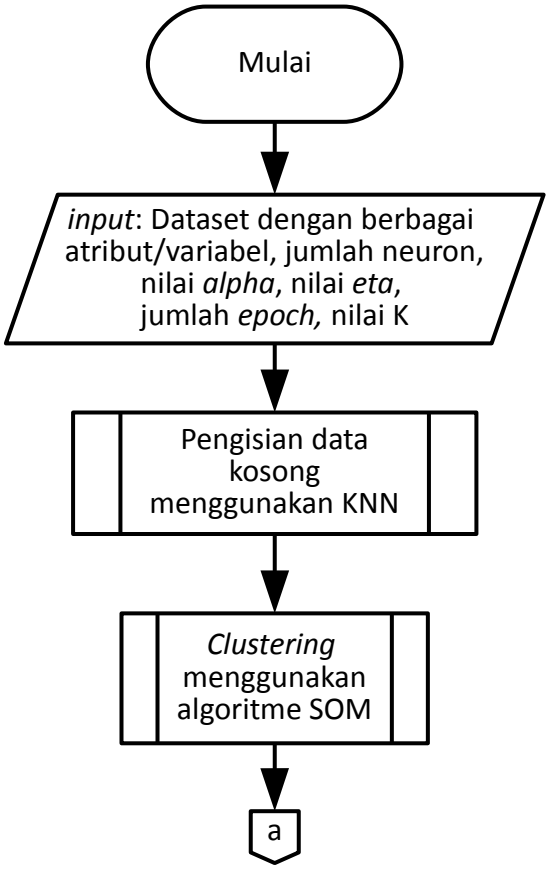
BAB 3 METODOLOGI PENELITIAN

3.1 Tipe Penelitian

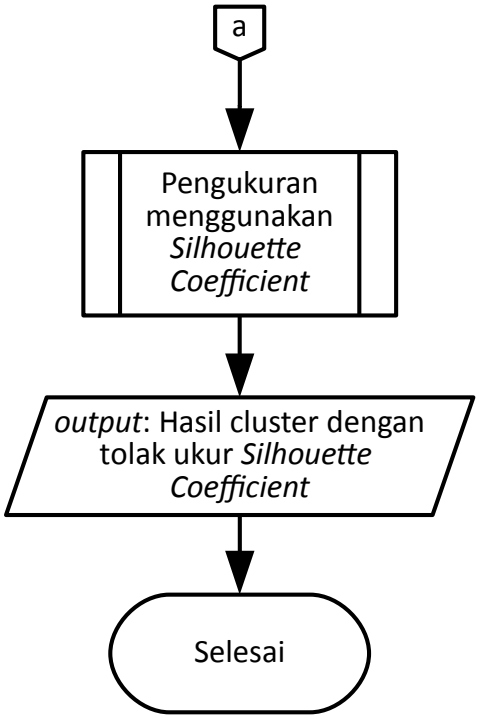
Tipe penelitian yang digunakan untuk menyelesaikan permasalahan yang ada tercantum pada rumusan masalah yaitu tipe nonimplementatif-analitik. Tipe penelitian nonimplementatif-analitik merupakan penelitian yang difokuskan pada penerapan berbagai metode atau algoritme dalam melakukan analisis permasalahan-permasalahan yang telah didefinisikan sebelumnya.

3.2 Metode Penelitian

Pada tipe penelitian non-implementatif analitik, metode dalam penyelesaian masalah diperoleh dengan melakukan sebuah perancangan model algoritme. Perancangan model dilakukan sebagai tahapan awal untuk mendesain metode yang akan diimplementasikan pada sistem. Perancangan model ini juga menjelaskan peran dari setiap algoritme yang akan digunakan pada penelitian. Model algoritme yang akan dibangun dapat dilihat pada Gambar 3.1 dan Gambar 4.1.



Gambar 3.1 Diagram alir metodologi secara umum



Gambar 3.1 Diagram alir metodologi secara umum (lanjutan)

Penjelasan dari metode yang akan dibangun adalah sebagai berikut:

3.2.1 Pengisian Data Kosong menggunakan KNN

K-Nearest Neighbors Missing-Value Imputation dilakukan untuk mengisi data yang tidak ada pada *dataset.* Pada latar belakang sebelumnya telah disebutkan bahwa banyak beberapa variabel atau atribut data yang tidak memiliki nilai. Pada penelitian ini, penulis menggunakan teknik pengisian data kosong menggunakan KNN dan menguji bagaimana pengaruh dari penggunaan algoritme tersebut pada proses *clustering*.

3.2.2 Clustering menggunakan Algoritme SOM

Algoritme SOM digunakan untuk melakukan proses pemetaan data PMKS dengan teknik *clustering*. Dalam hal ini, tiap-tiap neuron yang disusun pada algoritme SOM akan dianggap sebuah *cluster*. Sebuah neuron juga memiliki kemungkinan untuk tidak dianggap sebagai *cluster* jika, tidak ada anggota pada *dataset* yang masuk ke dalam neuron tersebut. Algoritme SOM yang dibangun memiliki ciri-ciri sebagai berikut:

1. *Dataset* dilakukan proses normalisasi menggunakan *min-max* sebelum dilakukan proses *training.* Proses normalisasi bertujuan untuk menyamakan rentang data pada tiap-tiap komponen variabel PMKS.

2. Jenis algoritme SOM yang digunakan yaitu jaringan SOM dengan topologi *2D-Rectangular*.
3. Jenis fungsi ketetanggaan yang digunakan yaitu menggunakan fungsi *Gaussian Neighborhood.*
4. Fungsi *update learning rate* yang digunakan yaitu menggunakan *update learning rate* menurun secara linier berdasarkan bertambahnya *epoch*.
5. Fungsi *update* nilai eta yang digunakan yaitu menggunakan fungsi *update* eta yang semakin menurun secara eksponensial berdasarkan bertambahnya *epoch*.

3.2.3 Pengujian menggunakan *Silhouette Coefficient*

Silhouette Coefficient dapat digunakan menghitung kualitas dari sebuah objek data yang masuk dalam kelompok *cluster* tertentu. Namun selain hal tersebut, *Silhouette Coefficient* juga dapat menentukan secara global bagaimana kualitas dari proses pembentukan *cluster* dengan rata-rata *Silhouette Coefficient* setiap objek data pada sebuah *dataset*. Dalam penelitian ini, rata-rata *Silhouette Coefficient* akan digunakan di setiap pengujian parameter algoritme untuk mencari parameter yang terbaik. Parameter terbaik yang dicari adalah parameter yang memberikan nilai rata-rata *Silhouette Coefficient* tertinggi.

3.3 Lokasi Penelitian

Penelitian ini menggunakan data mentah yang sudah didapatkan dan siap diolah tanpa perlu melakukan observasi atau pengambilan data terlebih dahulu. Sehingga, penelitian dilakukan langsung di kampus dengan menggunakan perangkat berupa laptop pribadi milik peneliti. Sebuah PC atau Laptop yang akan digunakan untuk penelitian akan dipersiapkan terlebih dahulu lingkungan pengembangan dan penelitiannya. Konfigurasi lingkungan pengembangan dan penelitian menyesuaikan dengan teknologi dan bahasa pemrograman yang digunakan.

3.4 Metode Pengumpulan Data

Pada penelitian ini, data yang digunakan adalah data Penyandang Masalah Kebutuhan Sosial (PMKS) Provinsi Jawa Timur pada tahun 2016. Data ini diambil dari Dinas Komunikasi dan Informatika Provinsi Jawa Timur. Data yang digunakan berupa hubungan antara Kabupaten/Kota dengan variabel-variabel pada *dataset* PMKS. Selain itu juga, terdapat beberapa nilai yang kosong atau belum terisi untuk beberapa Kabupaten/Kota dalam sebuah variabel.

Jumlah data *input* yang digunakan pada penelitian ini adalah sebanyak 38 Kabupaten/Kota. Jumlah fitur data yang ada pada *dataset* PMKS yaitu sebanyak 27 variabel PMKS. Sedangkan untuk fitur data yang digunakan pada penelitian ini yaitu sebanyak 26 variabel PMKS dikarenakan ada 1 fitur data yang nilainya kosong semua untuk setiap kabupaten sehingga tidak diikutsertakan ke dalam

proses *clustering*. Fitur data yang kosong untuk semua Kabupaten dan kota tersebut adalah variabel Keluarga Fakir Miskin.

Detail lengkap mengenai data yang digunakan dapat dilihat pada Lampiran A: Dataset yang Digunakan.

3.5 Metode Analisis Data

Metode analisis data dilakukan dengan menguji beberapa parameter yang ada pada algoritme SOM maupun algoritme KNN. Pengujian parameter dilakukan untuk mengetahui sejauh mana metode yang dibangun dapat memecahkan masalah yang sudah dijelaskan sebelumnya. Pengujian ini terdiri dari dibagi menjadi 3 jenis yaitu pengujian pada algoritme SOM dan pengujian pada metode *missing-value imputation* serta analisis *cluster* yang tebentuk:

a) Pengujian algoritme SOM

   1. Pengujian konstanta alpha atau nilai *learning rate* dari proses *update* bobot pada algoritme SOM. Pengujian ini diukur menggunakan skala rata-rata *Silhouette Coefficient.*
   2. Pengujian konstanta eta atau nilai signifikansi *update* bobot neuron tetangga pada algoritme SOM. Pengujian ini juga diukur menggunakan skala rata-rata *Silhouette Coefficient*.
   3. Pengujian jumlah *epoch* pada algoritme SOM. Pengujian tersebut diukur menggunakan skala rata-rata *Silhouette Coefficient*.
   4. Pengujian jumlah neuron pada algoritme SOM. Pengujian ini diukur menggunakan skala rata-rata *Silhouette Coefficient*

b) Pengujian penggunaan KNN *missing-value imputation*

   1. Pengujian ini dilakukan untuk mengetahui pengaruh pengisian data kosong pada proses *clustering.* Sama seperti pada pengujian algoritme SOM, pengujian ini diukur menggunakan skala rata-rata *Silhouette Coefficient.*

c) Analisis *cluster* yang terbentuk

   1. Analisis *cluster* yang terbentuk yaitu mengambil hasil *clustering* dengan *Silhouette Coefficient* tertinggi. Kemudian hasil clustering tersebut dianalisis menggunakan perbedaan variabel (selisih) antar *cluster* yang terbentuk.

3.6 Peralatan Pendukung yang Digunakan

Tahap implementasi mengacu pada desain model yang telah dibuat pada subbab sebelumnya. Pada penelitian ini, algoritme SOM akan diimplementasikan pada perangkat keras dengan spesifikasi sebagai berikut:

Processor : Intel Core i5-7200U 2.5 Ghz

Memory	: 4GB DDR4 RAM
Storage	: 1 TB Hard disk drive (HDD)
GPU	: Nvidia Geforce 940MX

Sedangkan untuk perangkat lunak, algoritme SOM akan diimplementasikan dengan menggunakan beberapa *tools* dan bahasa pemrograman dengan detail sebagai berikut:

Bahasa pemrograman	: Python dan Javascript
Framework yang digunakan	: Flask dan p5.js
Editor	: Sublime Text

Bahasa pemrograman Python digunakan untuk mengimplementasikan model algoritme yang telah dirancang. Sedangkan bahasa pemrograman Javascript digunakan untuk menampilkan secara *real-time* kemajuan algoritme dalam format berupa halaman web pada saat algoritme SOM dijalankan. Untuk memudahkan dalam mengimplementasikan algoritme tersebut ke dalam sistem, maka *framework* Flask dan p5.js pada penelitian kali ini. *Framework* Flask merupakan *framework* yang digunakan untuk mengimplementasikan sebuah halaman *web* pada bahasa pemrograman Python. Sedangkan *framework* p5.js digunakan sebagai visualisasi proses *clustering* yang sedang berlangsung.

BAB 4 PERANCANGAN

4.1 Perancangan Algoritme

Perancangan algoritme adalah tahapan yang digunakan untuk memberi gambaran mengenai prosedur yang akan dilakukan sebelum nantinya diimplementasikan pada bahasa pemprograman. Pada subbab ini, rancangan sebuah algoritme akan digambarkan dengan diagram alir atau *flowchart.*

4.1.1 Deskripsi umum algoritme

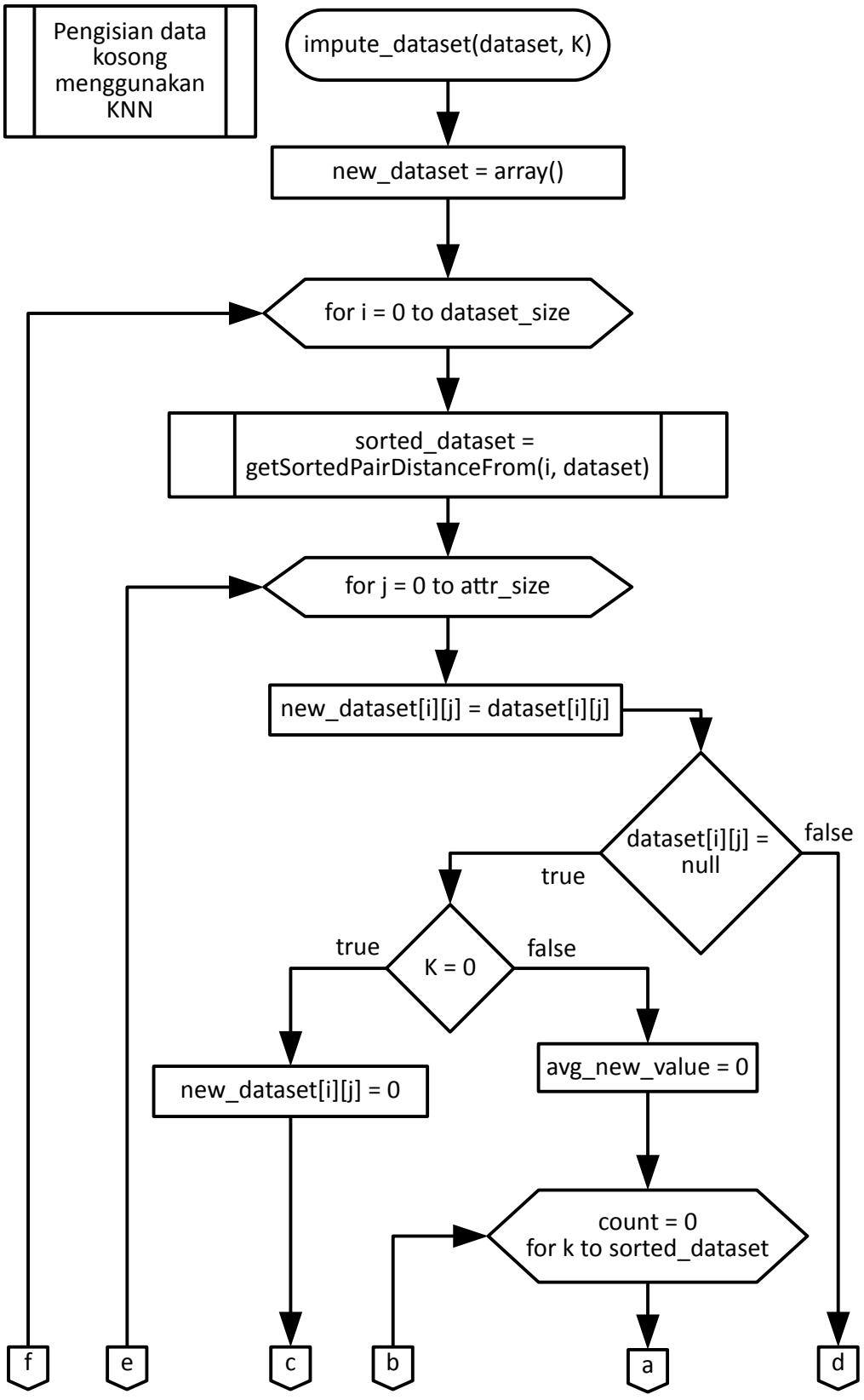
Subbab ini membahas mengenai penjelasan umum bagaimana algoritme bekerja untuk menyelesaikan masalah yang telah didefinisikan sebelumnya. Secara umum, algoritme yang akan dibangun memiliki fungsi dapat melakukan *clustering* dengan *output* berupa bentuk jaringan SOM yang digunakan beserta keanggotaan (dalam hal ini Kabupaten/Kota) pada neuron-neuron di jaringan SOM. Sehingga dari sebuah representasi berupa jaringan SOM dapat menjadi sebuah bentuk *cluster* yang dapat diukur menggunakan nilai *Silhouette Coefficient.*

Algoritme yang dibangun juga memiliki fungsi untuk dapat menyertakan teknik pengisian data menggunakan algoritme K-Nearest Neighbors. Pengguna dapat memilih untuk tidak proses pengisian data kosong sebelum melakukan proses *clustering* atau pengguna juga dapat memilih untuk melakukan proses pengisian nilai kosong dengan menyertakan nilai K yang akan digunakan.

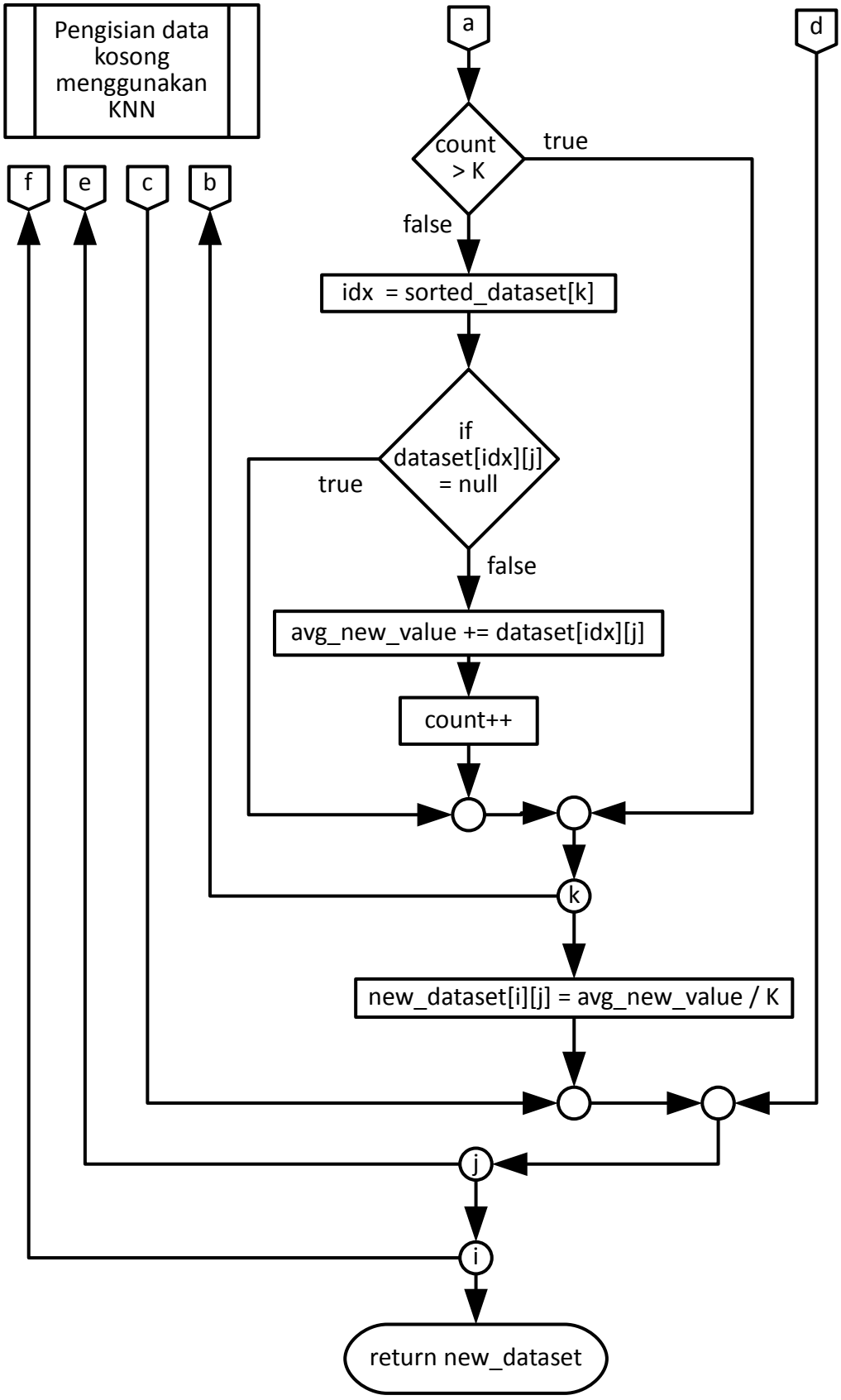
4.1.2 Pengisian *missing-value* menggunakan *K-Nearest Neighbors*

Pada pengisian *missing-value* menggunakan algoritme *K-Nearest Neighbors* (KNN), input yang dibutuhkan oleh algoritme adalah nilai K dan *dataset* yang memiliki nilai kosong itu sendiri. Masing-masing dari data kosong pada atribut tertentu akan dicari nilai dengan menggunakan K nilai yang memiliki jarak terdekat dengan data yang kosong tersebut. Kemudian dari K nilai yang diambil tersebut, rata-rata dari K nilai yang diambil tersebut akan dijadikan sebagai nilai yang baru. Diagram alir dari pengisian *missing-value* menggunakan KNN dapat dilihat pada Gambar 4.1.

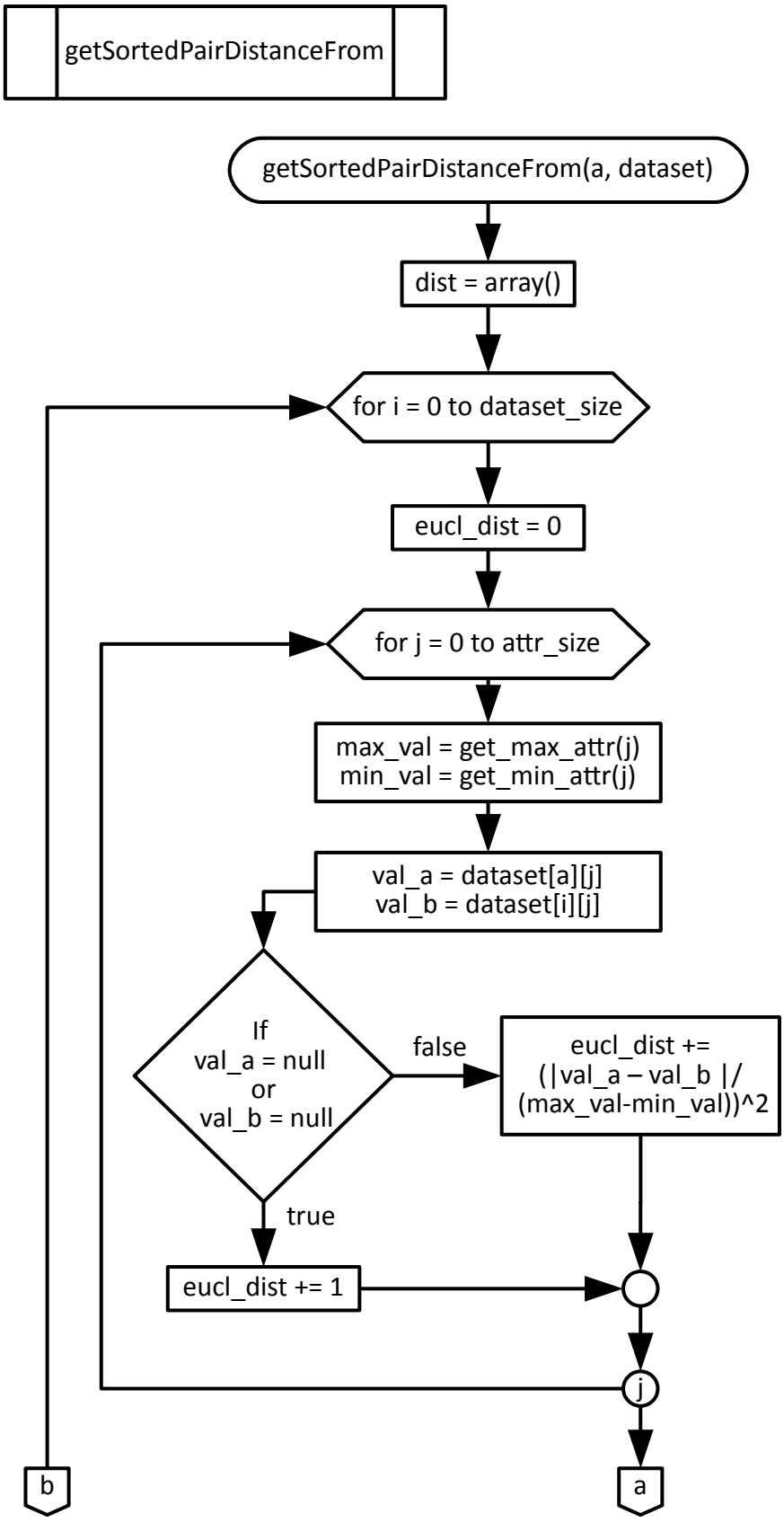
Masing-masing data *input* pada atribut yang nilainya kosong tersebut akan dicari jarak terdekatnya dengan data yang lain seperti yang dapat dilihat pada Gambar 4.2. Setelah didapatkan data yang memiliki jarak terdekat sebanyak K, nilai yang digunakan untuk mengisi data adalah rata-rata dari kandidat nilai sebanyak K pada atribut yang bersangkutan. Ketentuan kandidat nilai yang dipakai adalah nilai pada atribut yang bersangkutan tidak kosong juga.



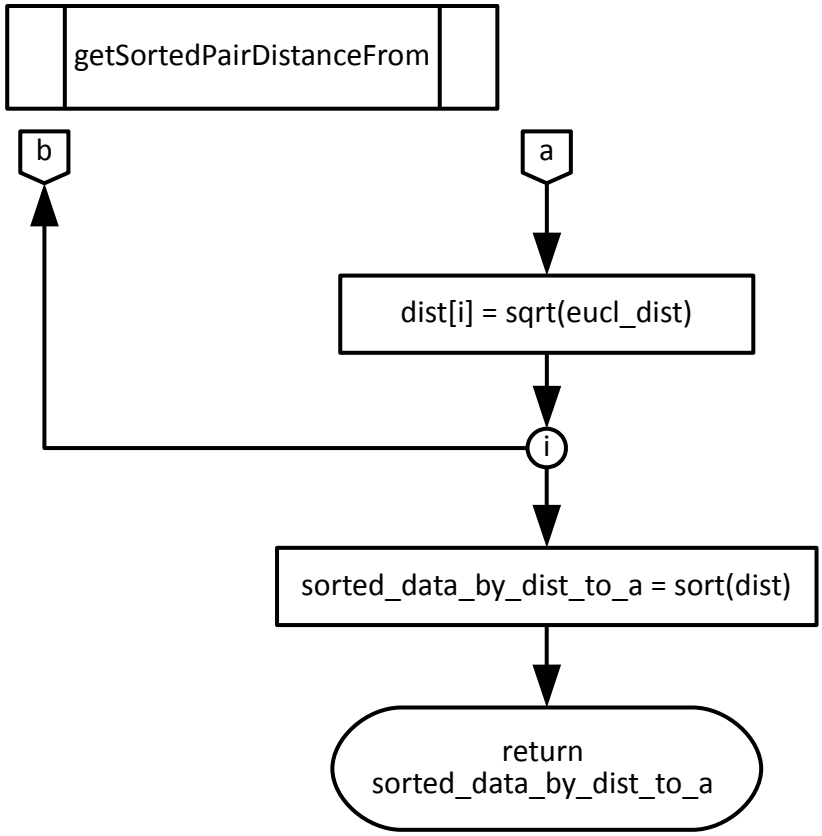
Gambar 4.1 Diagram alir pengisian data kosong menggunakan KNN



Gambar 4.1 Diagram alir pengisian data kosong menggunakan KNN (lanjutan)



Gambar 4.2 Diagram alir proses mendapatkan jarak tetangga terdekat



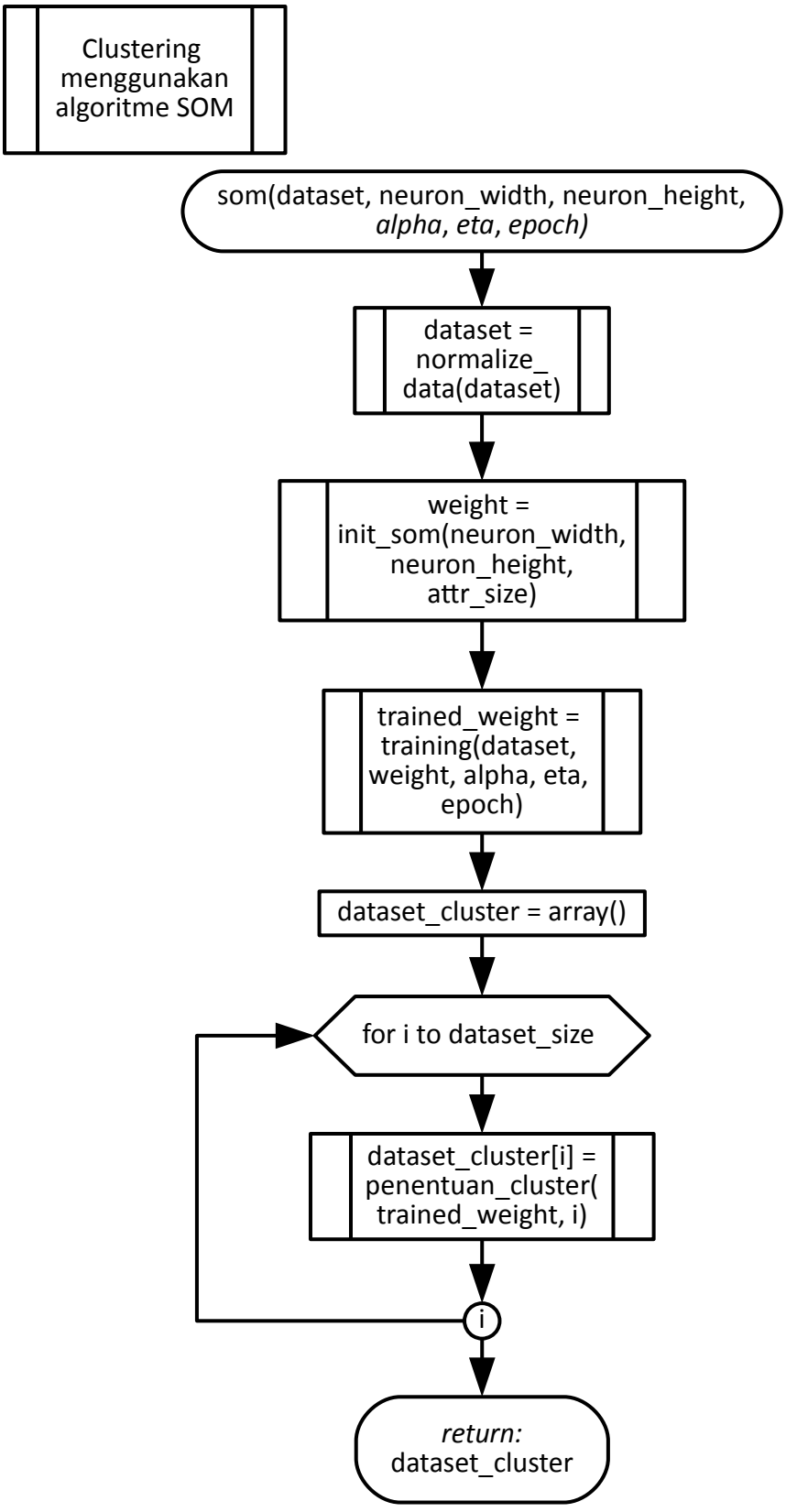
Gambar 4.2 Diagram alir proses mendapatkan jarak tetangga terdekat (lanjutan)

4.1.3 Tahapan umum pada algoritme SOM.

Input yang digunakan pada algoritme SOM yaitu *dataset input* yang terdiri dari berbagai atribut. Selain itu, parameter algoritme yang harus dimasukkan yaitu jumlah neuron, nilai *learning rate* atau *alpha*, nilai signifikansi *update* bobot neuron tetangga atau nilai *eta* dan jumlah *epoch*.

Secara umum, proses SOM diawali dengan melakukan normalisasi data terlebih dahulu. Kemudian sebelum algoritme SOM dijalankan, proses inisialiasi dilakukan untuk mengisi bobot awal jaringan dengan nilai random. Selanjutnya, proses training dilakukan sesuai dengan data *input*, nilai *alpha,* nilai *eta* serta jumlah *epoch* yang dimasukkan. Setelah proses *training* selesai, bobot jaringan hasil dari proses *training* tersebut digunakan untuk menentukan *cluster* atau melakukan proses *clustering*.

Diagram alir dari tahapan umum algoritme SOM dapat dilihat pada Gambar 4.3. Pada diagram tersebut, proses penentuan *cluster* dilakukan secara berulang untuk setiap objek data pada sebuah *dataset*. Nilai kembalian dari proses *clustering* dengan menggunakan algoritme SOM adalah hasil dari sebuah *cluster* yang terbentuk untuk masing-masing objek data di *dataset* yang digunakan.

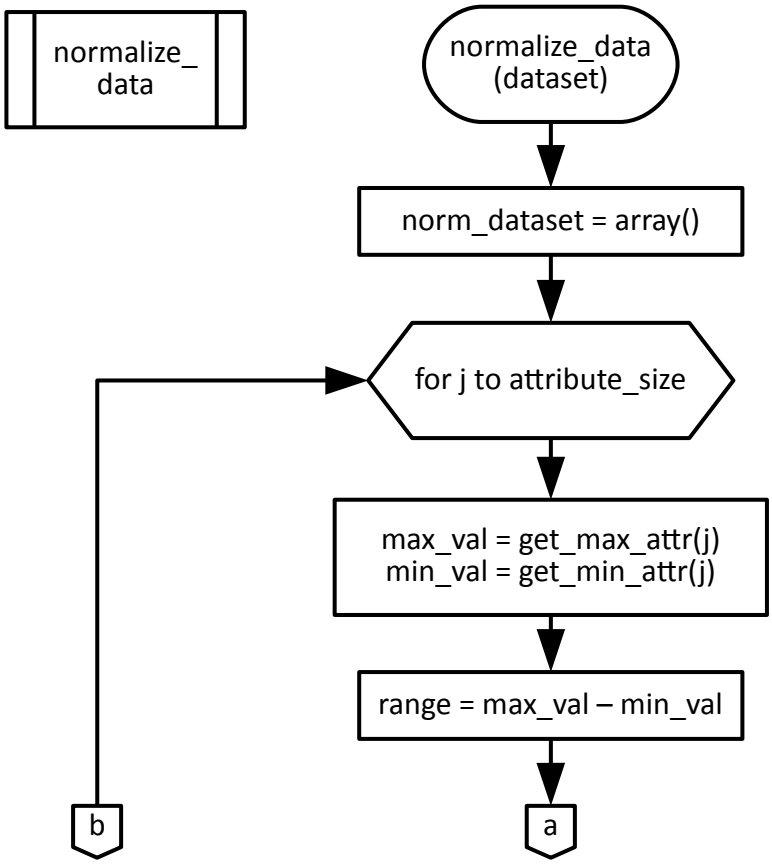


Gambar 4.3 Diagram alir algoritme SOM secara umum

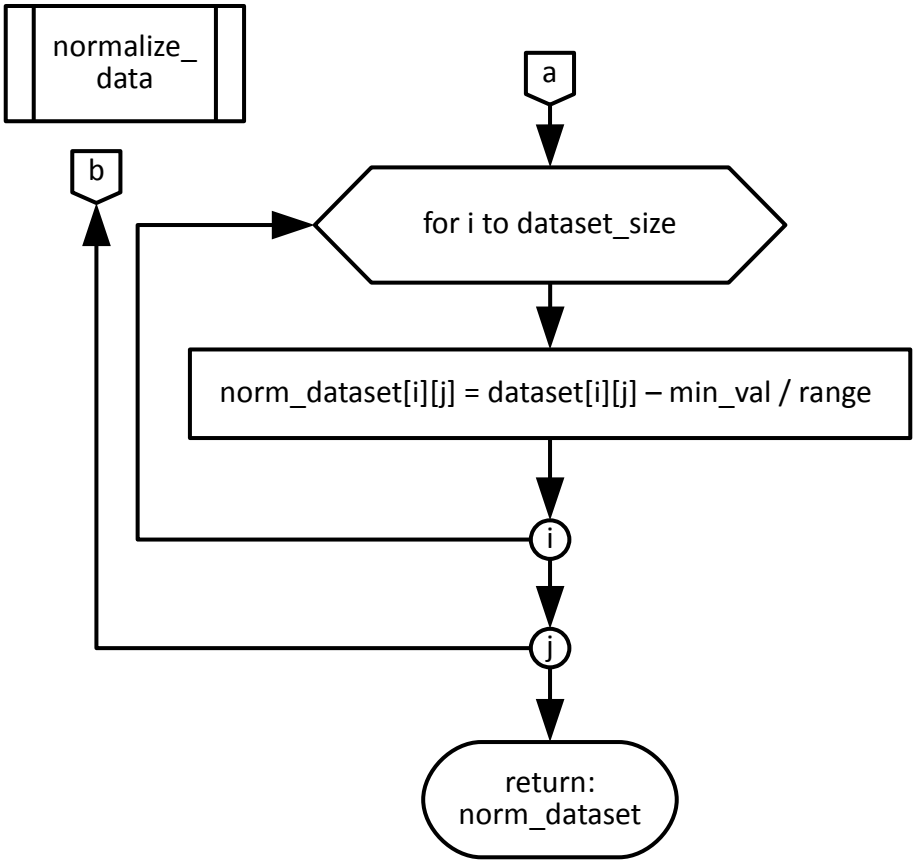
4.1.4 Normalisasi *min-max*

Normalisasi bertujuan untuk menyamakan nilai pada setiap atribut dengan rentang nilai yang sama. Sehingga, algoritme akan memperlakukan atribut pada *dataset* sama rata. Dalam hal ini, sama rata memiliki arti bahwa tidak ada atribut yang terlalu dominan dibanding lainnya ketika proses *training* telah dilakukan. Pada normalisasi *min-max*, nilai tertinggi dari sebuah atribut akan menjadi satu dan nilai terendah pada sebuah atribut akan berubah menjadi nol. Sedangkan untuk nilai lainnya, nilai yang berada di antara nilai tertinggi pada atribut dan nilai terendah pada atribut akan dipetakan atau dijadikan skala berdasarkan acuan nilai tertinggi dan terendah atribut, namun dengan rentang nilai antara 0 hingga 1.

Secara umum, normalisasi data *min-max* memiliki alur seperti pada Gambar 4.4. Hal yang dilakukan pertama yaitu mengambil nilai maksimal dan minimal pada atribut. Kemudian nilai maksimal dan minimal tersebut digunakan untuk menentukan *range* atribut. Nilai *range* variabel dapat dicari dengan mengurangi nilai maksimal variabel dengan nilai minimal pada atribut. Data yang dinormalisasi dengan *min-max* dapat dapat dihitung dengan mengurangi nilai data asli dengan nilai minimal pada variabel yang terpilih kemudian membaginya dengan nilai *range* pada variabel yang terpilih.



Gambar 4.4 Diagram alir normalisasi data

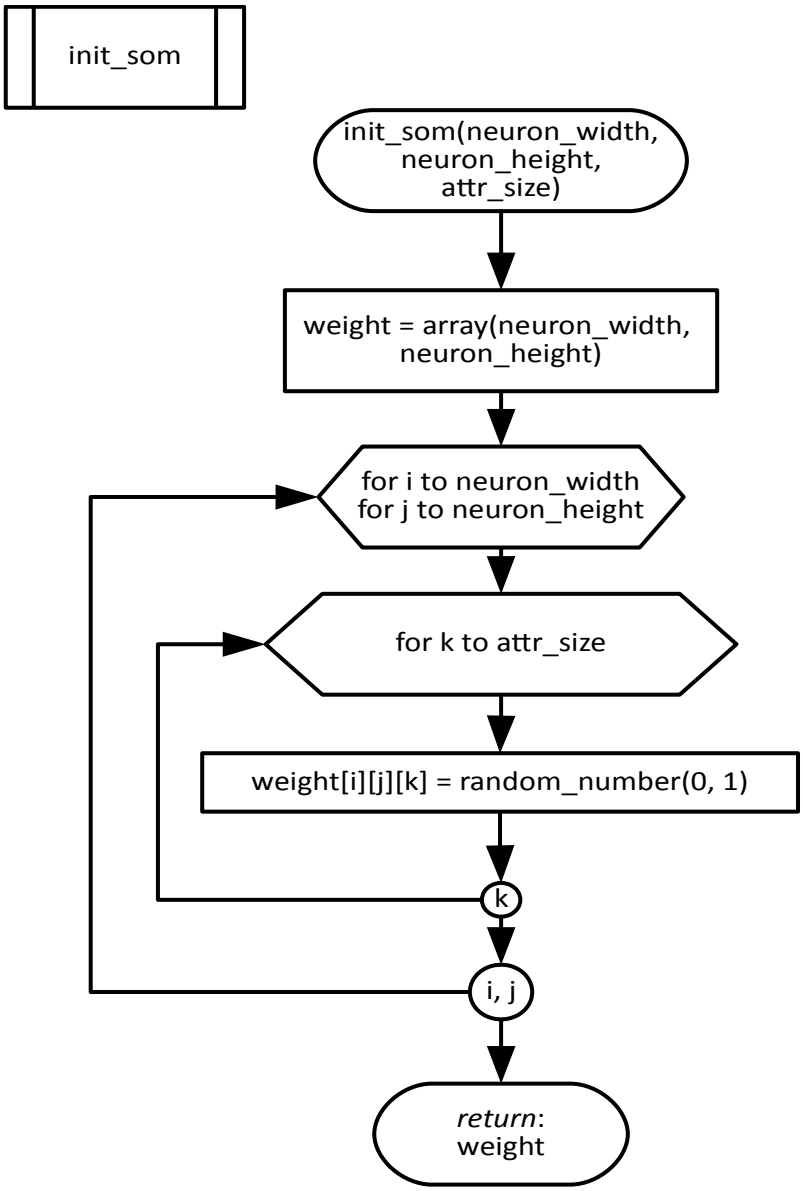


Gambar 4.4 Diagram alir normalisasi data (lanjutan)

4.1.5 Inisialisasi bobot awal jaringan *Self-Organizing Maps*

Sebelum dapat digunakan dalam proses *training*, sebuah jaringan SOM harus diinisialisasi terlebih dahulu. Jaringan SOM akan diinisialisasi sesuai dengan ukuran sesuai dengan jumlah atribut yang digunakan dan jumlah neuron *output* yang ditentukan oleh pengguna. Umumnya, jaringan SOM direpresentasikan dalam bentuk *array* atau *list* sesuai dengan atribut dan jumlah neuron *output* yang ditentukan pengguna. Sedangkan nilai-nilai bobot jaringan SOM diinisialiasi dengan nilai *random* antara 0 hingga 1.

Proses inisialisasi bobot awal jaringan dapat dilihat pada Gambar 4.5. *Input* fungsi dari diagram alir tersebut adalah jumlah neuron yang direpresentasikan sebagai lebar dan tinggi neuron serta jumlah *attribut* pada *dataset* yang digunakan. *Output* fungsi dari diagram alir tersebut adalah *array* berisi nilai bobot yang sesuai dengan *input* fungsi.



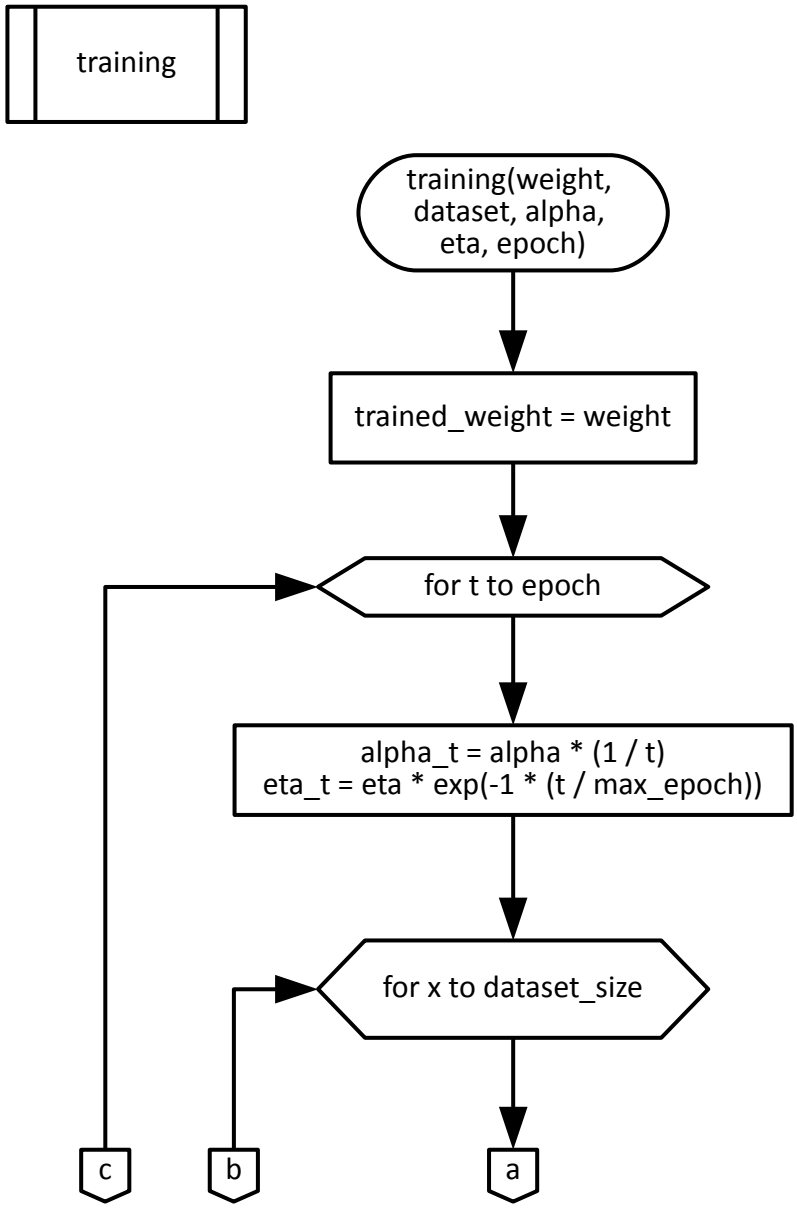
Gambar 4.5 Diagram alir inisialisasi bobot awal jaringan SOM

4.1.6 Proses *training*

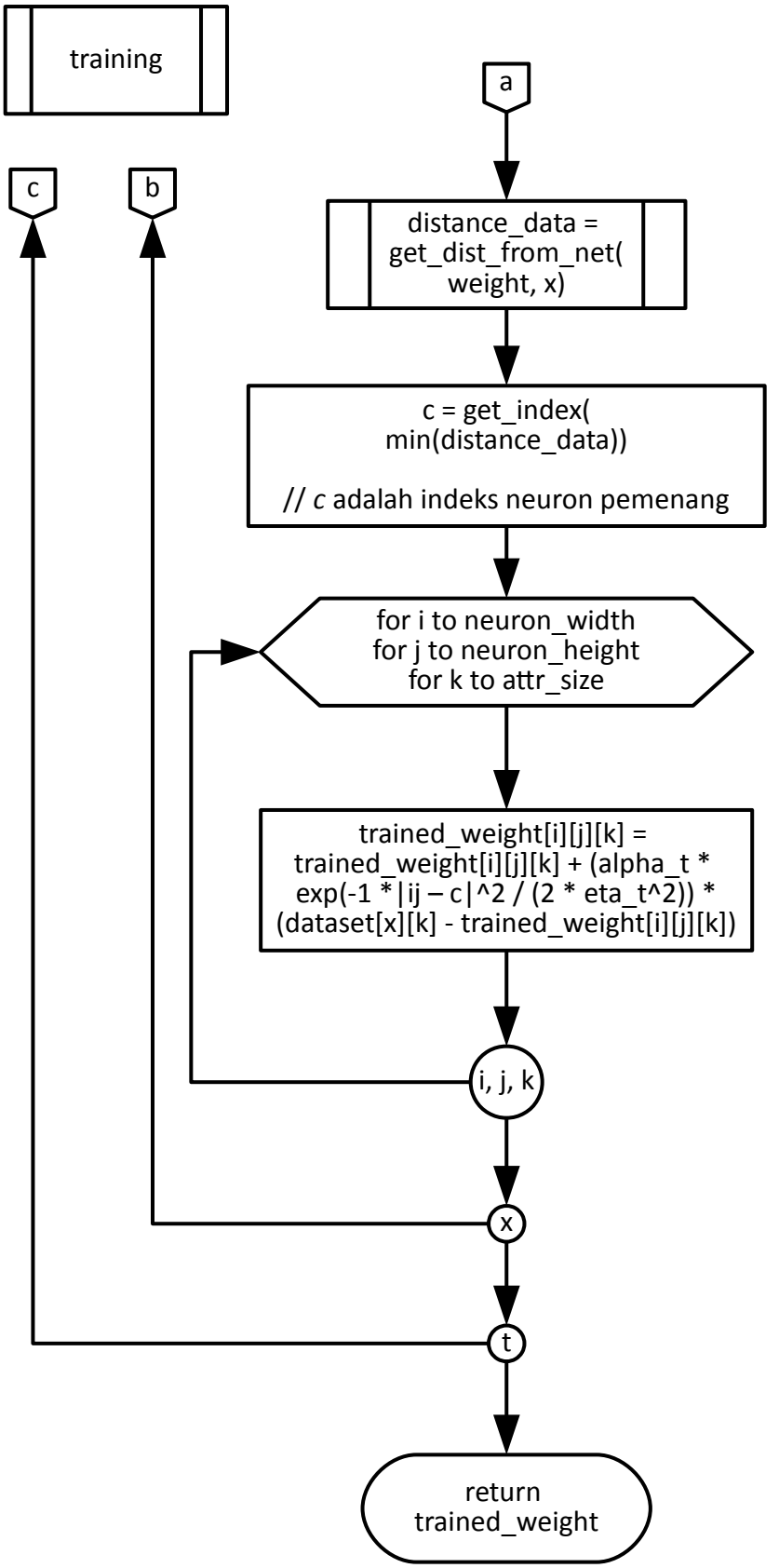
Proses *training* bertujuan untuk menyesuaikan nilai-nilai bobot jaringan pada SOM agar dapat digunakan dalam proses *clustering* secara tepat. Secara umum, proses *training* pada jaringan SOM terdiri dari 2 tahap, yaitu tahap penentuan *best-matching unit* (BMU) atau neuron pemenang dan proses adaptasi atau *update* bobot. Proses penentuan neuron pemenang dilakukan dengan mencari neuron yang memiliki jarak terkecil dengan data *input* yang diproses. Kemudian, proses *update* bobot akan memperbarui nilai bobot jaringan sesuai dengan neuron pemenang yang telah ditentukan. Pada penelitian ini, proses *update*

bobot jaringan menggunakan fungsi ketetanggaan *Gaussian*. Proses ini dilakukan berulang kali untuk setiap data *input* dan juga dilakukan berulang kali sesuai dengan jumlah *epoch* yang telah ditentukan. Pada setiap akhir satu *epoch* atau ketika semua data *input* telah diproses, dilakukan sebuah proses *update* nilai konstanta *learning rate* dan nilai *eta*.

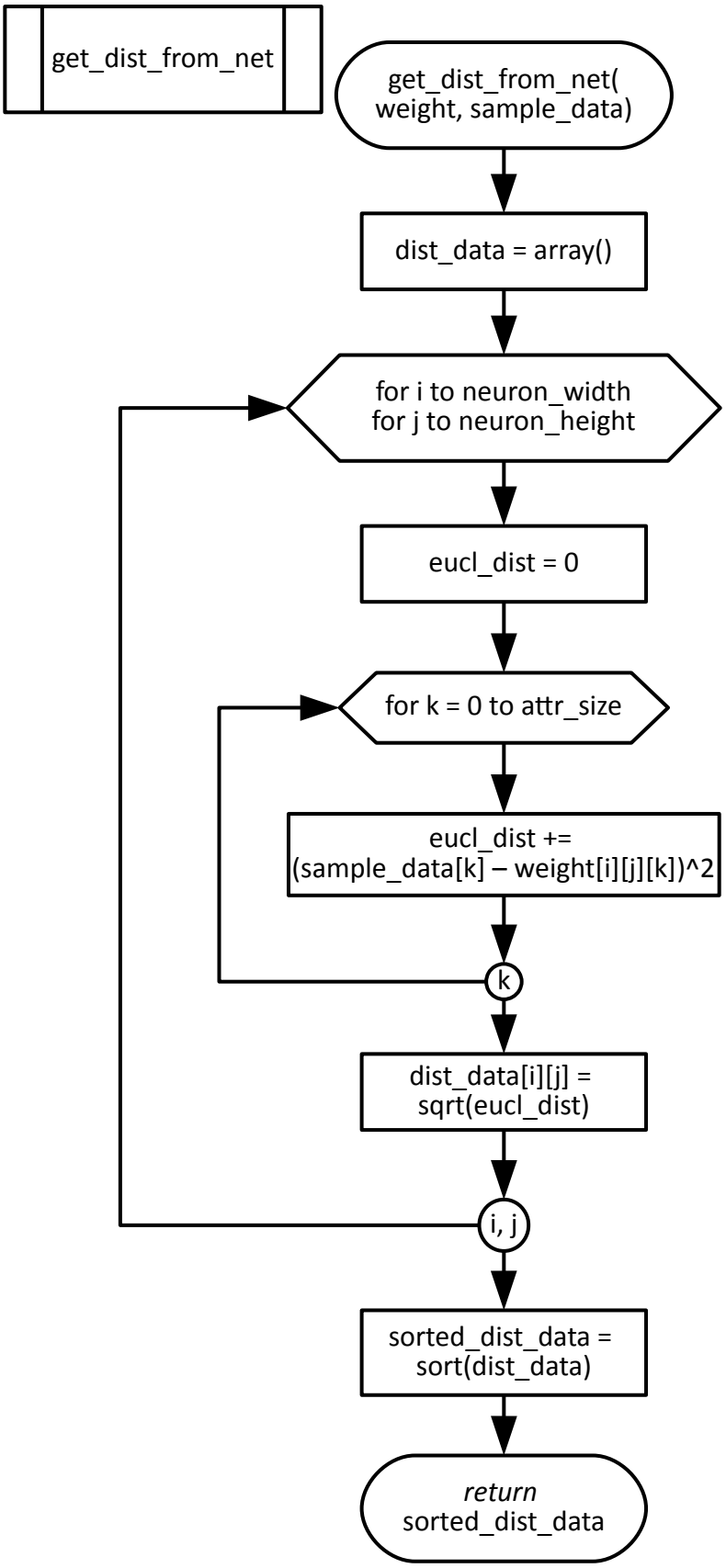
Diagram alir proses *training* dapat dilihat pada Gambar 4.6. Sedangkan, diagram alir untuk proses penentuan *best-matching unit* (BMU) dapat dilihat pada Gambar 4.7.



Gambar 4.6 Diagram alir proses *training* jaringan SOM



Gambar 4.6 Diagram alir proses *training* jaringan SOM (lanjutan)

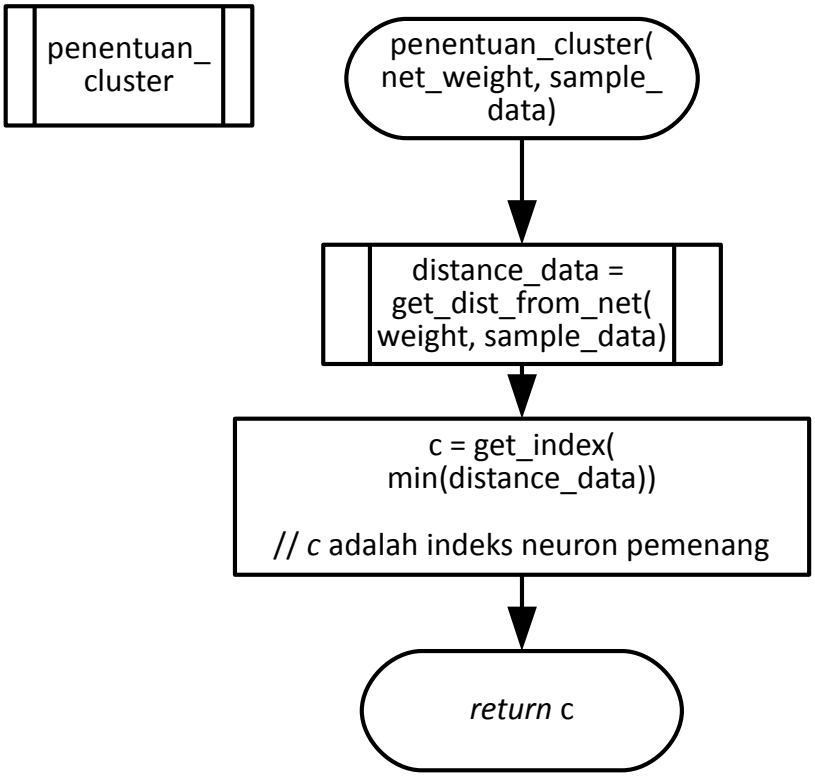


Gambar 4.7 Diagram alir proses penentuan *best-matching unit* (BMU)

4.1.7 Proses penentuan *cluster* atau *clustering*

Proses penentuan *cluster* pada dasarnya yaitu proses menentukan *best-matching unit* (BMU) atau neuron pemenang. Proses ini dilakukan dengan mencari jarak terkecil antara neuron-neuron pada jaringan SOM dengan masing-masing data *input*. Sehingga, setiap data *input* nanti akan memiliki satu neuron pemenang yang dapat dikatakan sebuah *cluster*.

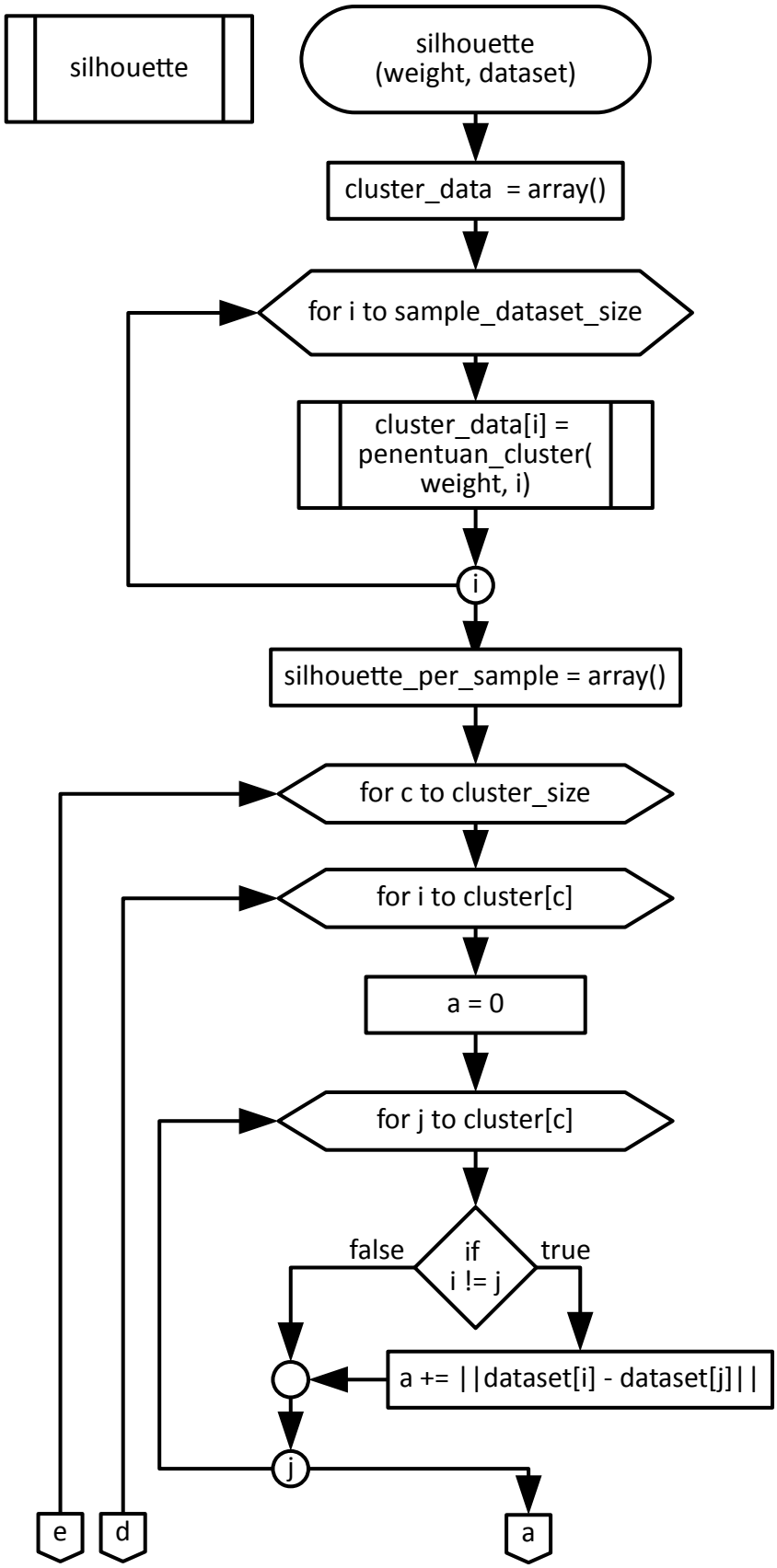
Proses penentuan *cluster* ini menggunakan jaringan SOM yang telah di-*training*. Diagram alir dari proses *training* dapat dilihat pada Gambar 4.8. Pada gambar tersebut, *input* dari algoritme berupa dataset yang akan dicari clusternya beserta bobot jaringan yang telah di-*training*. Output akhir dari proses penentuan cluster berupa keanggotaan *cluster* dari data input.



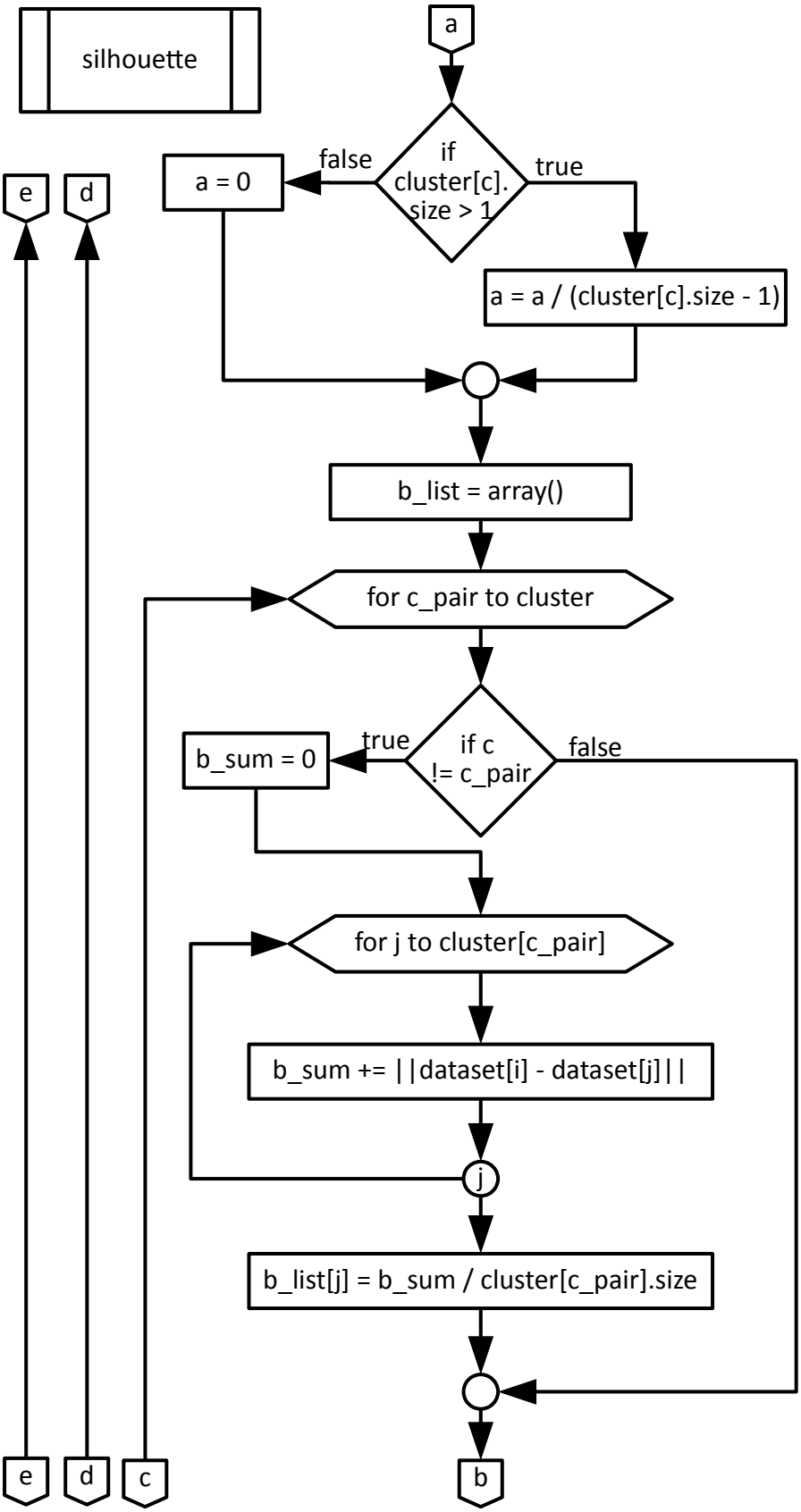
Gambar 4.8 Diagram alir proses penentuan *cluster* sebuah objek data

4.1.8 Perhitungan nilai *Silhouette Coefficient*

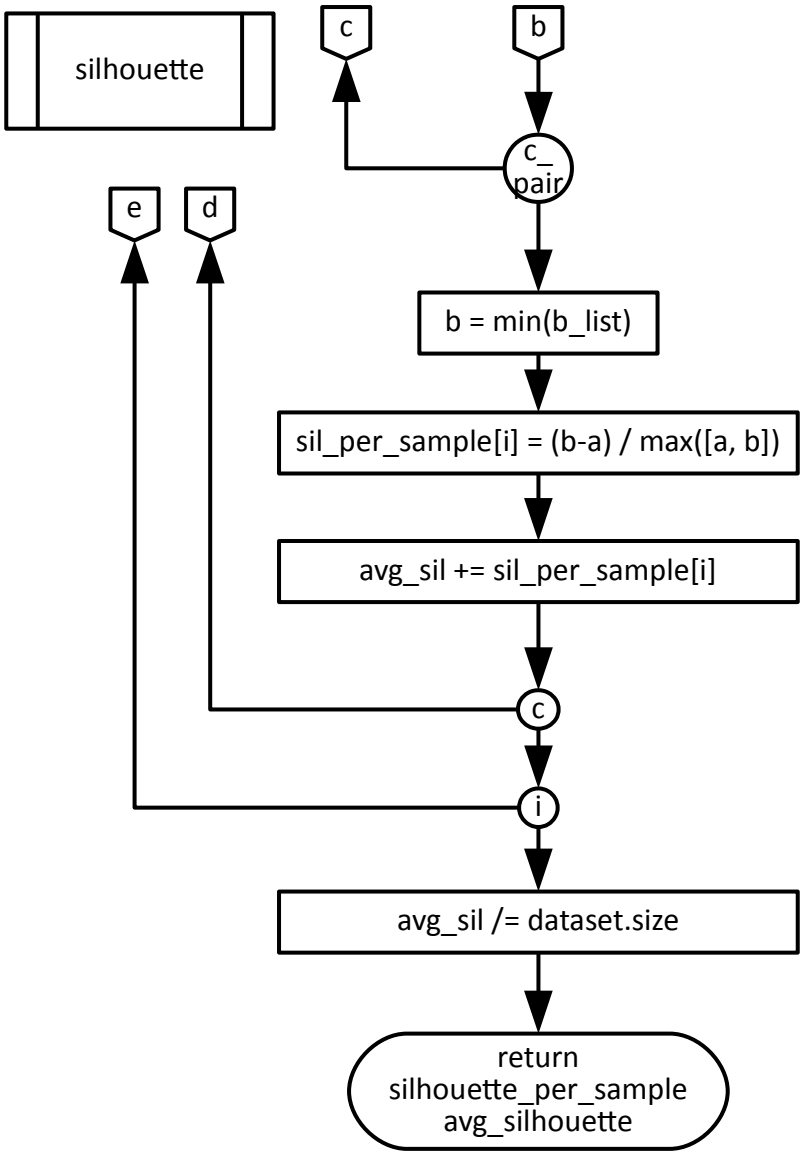
Setelah proses *training* dan *clustering* dilakukan, proses perhitungan nilai *Silhouette Coefficient* dilakukan untuk mengetahui kualitas dari sebuah *cluster* yang terbentuk. Proses perhitungan dilakukan untuk masing-masing objek data pada dataset. Secara umum, proses ini dilakukan dengan menghitung jarak rata-rata sebuah data dengan data lain pada satu *cluster* yang sama. Kemudian, langkah selanjutnya yaitu menghitung jarak data terpendek dengan *cluster* lain yang terbentuk. Diagram alir dari proses perhitungan nilai *Silhouette Coefficient* dapat dilihat pada Gambar 4.9, Gambar 6.2 dan Gambar 6.3.



Gambar 4.9 Diagram alir perhitungan *Silhouette Coefficient*



Gambar 4.9 Diagram alir perhitungan *Silhouette Coefficient* (lanjutan)



Gambar 4.9 Diagram alir perhitungan *Silhouette Coefficient* (lanjutan)

4.2 Perhitungan Manual

Tahapan perhitungan manual berfungsi sebagai validasi algoritme ketika telah diimplementasikan ke sistem. Harapannya, agar algoritme yang dibangun nantinya dapat sesuai dengan tahap perhitungan manual yang telah dilakukan. Pada tahap ini, akan digunakan sebagian dari *dataset* sebagai data *dummy* dalam melakukan perhitungan manual.

4.2.1 Pengisian *missing-value* menggunakan *K-Nearest Neighbors*

Pada tahap ini, data yang kosong pada atribut tertentu akan diisi menggunakan nilai-nilai dari data lain yang memiliki jarak terdekat dengan sebuah data *input*. Jarak yang digunakan sebagai pendekatan pengisian nilai

yaitu jarak pada sebuah atribut di *dataset* yang nilainya lengkap untuk seluruh data *input*. Sebagai contoh perhitungan, *dataset* yang akan digunakan dapat dilihat pada Tabel 4.1. Dalam hal ini, Nilai K yang akan digunakan untuk proses normalisasi adalah 3.

Tabel 4.1 Dataset yang digunakan sebagai contoh perhitungan pengisian data kosong

No	Kabupaten/ Kota	Anak Jalanan	Tuna Susila	Pengemis	Gelandangan dan Gelandangan Psikotik	Anak Balita Terlantar	Anak Terlantar
1	Pacitan	(kosong)	(kosong)	6	32	(kosong)	(kosong)
2	Ponorogo	52	(kosong)	153	114	562	983
3	Trenggalek	16	(kosong)	68	42	4471	19633
4	Tulungagung	6	162	32	13	87	323
5	Blitar	46	(kosong)	88	75	94	188
6	Kediri	184	107	154	132	8	406
7	Malang	210	(kosong)	220	139	57	4788
8	Lumajang	15	116	97	6	21	147
9	Jember	171	172	332	265	142	3062
10	Banyuwangi	(kosong)	(kosong)	297	267	333	1108

Untuk pengisian data pada Kabupaten Pacitan, variabel Anak Jalanan dapat dilakukan sebagai berikut:

1. Menghimpun semua kemungkinan jarak data yang tersedia antara Kabupaten Pacitan dengan kabupaten/kota yang nilainya tidak kosong pada atribut/variabel Anak Jalanan. Dalam hal ini kab/kota yang terpilih dapat dilihat pada Tabel 4.2:

Tabel 4.2 Himpunan jarak antara Kab. Pacitan dengan Kab/Kota yang nilainya tidak kosong

No.	Himpunan Jarak
1	d(pacitan, ponorogo)
2	d(pacitan, trenggalek)
3	d(pacitan, tulungagung)
4	d(pacitan, blitar)
5	d(pacitan, kediri)

Tabel 4.2 Himpunan jarak antara Kab. Pacitan dengan Kab/Kota yang nilainya tidak kosong (lanjutan)

No.	Himpunan Jarak
6	d(pacitan, malang)
7	d(pacitan, lumajang)
8	d(pacitan, jember)

2. Menghitung jarak Euclidian antara Kabupaten Pacitan dengan daerah lain. Sedangkan dalam pemilihan atribut untuk jarak Euclidian yaitu mengabaikan atribut yang memiliki *missing value*. Dalam kasus ini, atribut Pengemis, Gelandangan dan Gelandangan Psikotik yang digunakan untuk penghitungan jarak Euclidian. Jarak Euclidian yang digunakan dalam penghitungan jarak yaitu jarak Euclidian yang sudah dinormalisasi menggunakan *min-max*:

   Jarak normalisasi antara Kabupaten Pacitan dan Ponorogo untuk atribut pengemis:

$$\begin{aligned} d_{pengemis}(a,\ b) &= \frac{|X_b - X_a|}{max(pengemis) - min(pengemis)} \\ &= \frac{|153-6|}{332-6} \\ &= 0.45092 \end{aligned}$$

   Sedangkan jarak normalisasi antara Kabupaten Pacitan dan Ponorogo untuk atribut gelandangan dan gelandangan psikotik adalah sebagai berikut:

$$\begin{aligned} d_{gelpsikotik}(a,\ b) &= \frac{|X_b - X_a|}{max(gelpsikotik) - min(gelpsikotik)} \\ &= \frac{|114-32|}{267-6} \\ &= 0.31418 \end{aligned}$$

   Karena pada atribut Anak Jalanan, Tuna Susila, Anak Balita Terlantar dan Anak Terlantar terdapat salah satu Kabupaten/Kota yang kosong antara Pacitan dan Ponorogo, maka jarak untuk masing-masing atribut tersebut diberi nilai 1.

$$\begin{aligned} d_{anakjalanan}\ (a,\ b) &= 1 \\ d_{tunasusila}\ (a,\ b) &= 1 \\ d_{balitaterlantar}\ (a,\ b) &= 1 \\ d_{anakterlantar}\ (a,\ b) &= 1 \end{aligned}$$

   Sehingga jarak Euclidian antara Kabupaten Pacitan dan Ponorogo adalah sebagai berikut:

$$d(a,\ b) = \sqrt{d_{aj}(a,\ b)^2+d_{ts}(a,\ b)^2+d_p(a,\ b)^2+d_{gp}(a,\ b)^2+\ldots}$$
$$\sqrt{\ldots\ d_{bt}(a,\ b)^2+d_{at}(a,\ b)^2}$$
$$= \sqrt{1^2+1^2+0.45092^2+0.31418^2+1^2+1^2}$$
$$= 2.0741349$$

Hasil perhitungan jarak Euclidian pada 7 Kabupaten/Kota dapat dilihat pada Tabel 4.3:

Tabel 4.3 Hasil perhitungan jarak antara Kab. Pacitan dengan Kab/Kota lain

No.	Himpunan Jarak	Jarak Euclidian
1	d(pacitan, ponorogo)	2,07413494757819
2	d(pacitan, trenggalek)	2,00938745607355
3	d(pacitan, tulungagung)	2,00291292625085
4	d(pacitan, blitar)	2,02247672462383
5	d(pacitan, kediri)	2,08636105916046
6	d(pacitan, malang)	2,14452426255811
7	d(pacitan, lumajang)	2,02184155525399
8	d(pacitan, jember)	2,40768551635412

3. Mengurutkan kabupaten/kota berdasarkan jarak Euclidian dari kecil ke yang terbesar. Selanjutnya memilih sebanyak K untuk Kabupaten/Kota yang memiliki jarak paling kecil seperti yang ditunjukkan pada Tabel 4.4. Dalam kasus ini, sebanyak tiga Kabupaten/Kota yang dipilih, yaitu Tulungagung, Trenggalek dan Lumajang.

Tabel 4.4 Memilih Kab/Kota sebanyak K yang ditentukan berdasarkan jarak terkecil

No	Himpunan Jarak	Jarak Euclidian
1	d(pacitan, tulungagung)	2,00291292625085
2	d(pacitan, trenggalek)	2,00938745607355
3	d(pacitan, lumajang)	2,02184155525399
4	d(pacitan, blitar)	2,02247672462383

Tabel 4.4 Memilih Kab/Kota sebanyak K yang ditentukan berdasarkan jarak terkecil (lanjutan)

No	Himpunan Jarak	Jarak Euclidian
5	d(pacitan, ponorogo)	2,07413494757819
6	d(pacitan, kediri)	2,08636105916046
7	d(pacitan, malang)	2,14452426255811
8	d(pacitan, jember)	2,40768551635412

4. Tiga Kab/Kota yang terpilih berfungsi sebagai kandidat yang akan menjadi nilai untuk mengisi data Kab. Pacitan atribut data Anak jalanan seperti yang ditunjukkan pada Tabel 4.5. Nilai yang digunakan untuk mengisi data kosong tersebut adalah hasil rata-rata dari tiga kandidat nilai tersebut.

Tabel 4.5 Tiga nilai pada atribut anak jalanan yang digunakan untuk pengisian data kosong

No	Kab/Kota	Anak Jalanan
1	Tulungagung	6
2	Trenggalek	16
3	Lumajang	15

Perhitungan nilai yang akan mengisi data Kab. Pacitan atribut Anak jalanan adalah sebagai berikut:

$$\widetilde{X}_j = \frac{1}{K}\sum_{k=1}^{K} v_{kj}$$

$$X_{pacitan} = \frac{1}{K}\left(X_{tulungagung} + X_{trenggalek} + X_{lumajang}\right)$$

$$= \frac{1}{3}(6+16+15)$$

$$= 12.333$$

Dengan demikian nilai yang digunakan untuk mengisi nilai pada data Kab. Pacitan atribut data Anak Jalanan adalah 12.333.

5. Jika kondisi dari kandidat nilai yang akan dipilih terdapat data yang kosong juga, maka nilai yang dipilih adalah data yang tidak kosong namun tetap memperhatikan urutan jarak Euclidian. Sebagai contoh yaitu pada pengisian data untuk Kab. Ponorogo dengan atribut data Tuna Susila yang dapat dilihat pada Tabel 4.6. Dalam hal ini, Kabupaten/Kota sudah diurutkan sesuai jarak Euclidian dan dipilih tiga kab/kota yang menjadi kandidat nilai, yaitu Blitar, Lumajang dan Tulungagung. Namun karena

data Blitar untuk atribut Tuna Susila juga kosong, maka data yang dipilih selanjutnya yaitu Kediri sebagai gantinya.

Tabel 4.6 Skenario pengisian nilai jika pada K jarak yang dipilih ada yang nilainya kosong

No.	Kab/Kota	Tuna Susila	Jarak Euclidian ke Kabupaten Ponorogo
1	Blitar	(kosong)	1,03711549330989
2	Lumajang	116	1,11810708991569
3	Tulungagung	162	1,16225328502167
4	Kediri	107	1,19989010995468
5	Malang	(kosong)	1,30468870538761
6	Jember	172	1,41305304635485
7	Banyuwangi	(kosong)	1,59418539673269
8	Trenggalek	(kosong)	1,69068203529452
9	Pacitan	(kosong)	2,07413494757819

4.2.2 Normalisasi *min-max* pada dataset

Dataset yang digunakan sebagai contoh untuk proses normalisasi dapat dilihat pada Tabel 4.7.

Tabel 4.7 Contoh *dataset* untuk proses normalisasi

No	Kabupaten/Kota	Anak Jalanan	Pengemis	Gelandangan dan Gelandangan Psikotik	Anak Terlantar
1	Pacitan	12.333	6	32	6701
2	Ponorogo	52	153	114	983
3	Trenggalek	16	68	42	19633
4	Tulungagung	6	32	13	323
5	Blitar	46	88	75	188
6	Kediri	184	154	132	406
7	Malang	210	220	139	4788
8	Lumajang	15	97	6	147
9	Jember	171	332	265	3062

Tabel 4.7 Contoh *dataset* untuk proses normalisasi (lanjutan)

No	Kabupaten/Kota	Anak Jalanan	Pengemis	Gelandangan dan Gelandangan Psikotik	Anak Terlantar
10	Banyuwangi	188.333	297	267	1108

Sebagai contoh, data Kabupaten Pacitan dengan atribut anak jalanan, data Kabupaten Trenggalek dengan atribut anak terlantar dan data Kabupaten Lumajang dengan atribut pengemis digunakan untuk proses normalisasi. Hal pertama yang dilakukan sebelum proses normalisasi data yaitu mencari nilai terbesar, terkecil dan *range* atribut dari data yang akan dinormalisasi. Nilai *range* adalah selisih nilai antara nilai terbesar dan nilai terkecil pada atribut data.

- Nilai maksimal, minimal dan *range* dari atribut anak jalanan

$$max(anakjalanan) = 210$$
$$min(anakjalanan) = 6$$
$$range(anakjalanan) = 210-6 = 204$$

- Nilai maksimal, minimal dan *range* dari atribut anak terlantar

$$max(anakterlantar) = 19633$$
$$min(anakterlantar) = 147$$
$$range(anakterlantar) = 19633-147 = 19486$$

- Nilai maksimal, minimal dan *range* dari atribut pengemis

$$max(pengemis) = 332$$
$$min(pengemis) = 6$$
$$range(pengemis) = 332-6 = 326$$

Perhitungan normalisasi untuk data Kabupaten Pacitan pada atribut anak jalanan yaitu

$$v'_{pacitan}(anakjalanan) = \frac{v_{pacitan}(anakjalanan)-min(anakjalanan)}{max(anakjalanan)-min(anakjalanan)} = \frac{v_{pacitan}(anakjalanan)-min(anakjalanan)}{range(anakjalanan)} = \frac{12.333-6}{204} = 0.0310458$$

Perhitungan normalisasi untuk data Kabupaten Trenggalek pada atribut anak terlantar yaitu

$$
\begin{aligned}
v'_{trenggalek}(anakterlantar) &= \frac{v_{trenggalek}(anakterlantar) - min(anakterlantar)}{max(anakterlantar) - min(anakterlantar)} \\
&= \frac{v_{trenggalek}(anakterlantar) - min(anakterlantar)}{range(anakterlantar)} \\
&= \frac{19633 - 147}{19486} \\
&= 1
\end{aligned}
$$

Perhitungan normalisasi untuk data Kabupaten Lumajang pada atribut pengemis yaitu

$$
\begin{aligned}
v'_{lumajang}(pengemis) &= \frac{v_{lumajang}(pengemis) - min(pengemis)}{max(pengemis) - min(pengemis)} \\
&= \frac{v_{lumajang}(pengemis) - min(pengemis)}{range(pengemis)} \\
&= \frac{97 - 6}{326} \\
&= 0.27919
\end{aligned}
$$

Dataset yang sudah dinormalisasi dapat dilihat pada Tabel 4.8. Warna hijau menunjukkan hasil dari ketiga contoh perhitungan manual yang telah dilakukan sebelumnya.

Tabel 4.8 Hasil akhir proses normalisasi data

No	Kabupaten/Kota	Anak Jalanan	Pengemis	Gelandangan dan Gelandangan Psikotik	Anak Terlantar
1	Pacitan	0,03104575	0	0,09961686	0,33634404
2	Ponorogo	0,2254902	0,45092025	0,4137931	0,0429026
3	Trenggalek	0,04901961	0,19018405	0,13793103	1
4	Tulungagung	0	0,0797546	0,02681992	0,00903213
5	Blitar	46	0,25153374	0,26436782	0,00210407
6	Kediri	184	0,45398773	0,48275862	0,01329159
7	Malang	1	0,65644172	0,50957854	0,23817099
8	Lumajang	15	0,2791411	0	0
9	Jember	171	1	0,99233716	0,14959458
10	Banyuwangi	188,333	0,89263804	1	0,04931746

4.2.3 Inisialisasi bobot awal

Pada kasus ini digunakan bobot jaringan dengan jumlah neuron *output* sebesar dengan arsitektur 2D-*Rectangular* dan jumlah atribut data yang digunakan sebanyak 3. Dalam implementasinya, bobot-bobot dalam jaringan SOM direpresentasikan sebagai matriks atau *array* sebanyak jumlah atribut data

yang digunakan jumlah neuron. Selain itu nilai awal yang digunakan untuk masing-masing bobot adalah bilangan acak antara 0 hingga 1. Dalam hal ini, jumlah atribut data sebanyak 3 sedangkan jumlah neuron sebanyak 4. Sehingga representasi bobot awal tersebut dapat dilihat pada Tabel 4.9.

Tabel 4.9 Inisialisasi bobot awal jaringan

	Atribut 1	**Atribut 2**	**Atribut 3**
Neuron 1,1	0,894055896375902	0,765182675079094	0,265319228617495
Neuron 1,2	0,173748888700865	0,445810777589486	0,161952424784464
Neuron 2,1	0,434879577568638	0,957322647806468	0,402313822293774
Neuron 2,2	0,228955391667413	0,022580102134164	0,14938963304644

4.2.4 Proses *training*

Proses *training* merupakan proses yang bertujuan untuk membuat bobot nilai pada jaringan SOM agar bernilai sesuai ketika digunakan dalam proses *clustering.* Sebagai contoh perhitungan manual, Diberikan dataset pada Tabel 4.10 untuk proses *training.* Sedangkan bobot jaringan awal yang digunakan adalah bobot jaringan yang ada pada Tabel 4.9.

Tabel 4.10 Contoh data *input* untuk proses *training*

	ANAK JALANAN	**GELANDANGAN DAN GELANDANGAN PSIKOTIK**	**ANAK TERLANTAR**
Pacitan	0,031045751633987	0,099616858237548	0,336344041876219
Jember	0,808823529411765	0,992337164750958	0,149594580724623
Banyuwangi	0,893790849673203	1	0,049317458688289

Sebagai contoh, diberikan nilai parameter awal jaringan SOM sebagai berikut:

$$\begin{aligned} maxepoch &= 3 \\ \alpha(0) &= 0.5 \\ \eta(0) &= 0.5 \end{aligned}$$

Pada epoch ke-1 nilai *learning rate alpha* dapat dihitung dengan persamaan berikut:

$$\alpha(t) = \alpha(0).\frac{1}{t}$$

$$\begin{aligned} \alpha(1) &= 0.5.\frac{1}{1} \\ &= 0.5 \end{aligned}$$

Sedangkan untuk konstanta *eta* atau nilai signifikansi perubahan bobot pada neuron tetangga dapat dihitung dengan persamaan:

$$\eta(t) = \eta(0).e^{(\frac{t}{T_{max}})}$$
$$\eta(1) = 0.5.e^{(\frac{1}{3})}$$
$$= 0.38526$$

Data *input* Kabupaten Pacitan adalah data pertama yang akan digunakan pada proses *training*. Proses yang pertama dilakukan terlebih dahulu yaitu menghitung jarak Euclidian antara data *input* Kabupaten Pacitan dengan masing-masing neuron.

$$d(pacitan, neuron_{11}) = \sqrt{((x_1 - w_{11.1})^2 + (x_2 - w_{11.2})^2 + (x_3 - w_{11.3})^2)}$$
$$= \sqrt{((0.031 - 0.894)^2 + (0.099 - 0.765)^2 + (0.336 - 0.265)^2)}$$
$$= \sqrt{(0.744 + 0.443 + 0.005)}$$
$$= 1.09215$$

$$d(pacitan, neuron_{12}) = \sqrt{((x_1 - w_{12.1})^2 + (x_2 - w_{12.2})^2 + (x_3 - w_{12.3})^2)}$$
$$= \sqrt{((0.031 - 0.174)^2 + (0.099 - 0.446)^2 + (0.336 - 0.162)^2)}$$
$$= \sqrt{(0.0204 + 0.1204 + 0.0302)}$$
$$= 0.41307$$

$$d(pacitan, neuron_{21}) = \sqrt{((x_1 - w_{21.1})^2 + (x_2 - w_{21.2})^2 + (x_3 - w_{21.3})^2)}$$
$$= \sqrt{((0.031 - 0.435)^2 + (0.099 - 0.957)^2 + (0.336 - 0.402)^2)}$$
$$= \sqrt{(0.1632 + 0.7361 + 0.0043)}$$
$$= 0.95031$$

$$d(pacitan, neuron_{22}) = \sqrt{((x_1 - w_{22.1})^2 + (x_2 - w_{22.2})^2 + (x_3 - w_{22.3})^2)}$$
$$= \sqrt{((0.031 - 0.229)^2 + (0.099 - 0.022)^2 + (0.336 - 0.149)^2)}$$
$$= \sqrt{(0.0392 + 0.0059 + 0.035)}$$
$$= 0.28293$$

Langkah selanjutnya adalah mencari jarak terkecil antara data *input* pada masing-masing neuron-neuron. Dalam hal ini jarak neuron 2,2 ke Kab. Pacitan merupakan jarak terkecil. Sehingga neuron pemenang pada proses *training* ini adalah neuron 2,2.

$$c_{pacitan} = \underset{i,j}{argmin}(\{1.09215, 0.41307, 0.95031, 0.28293\})$$
$$= [2,2]$$

Setelah neuron pemenang telah ditentukan, hal selanjutnya yang harus dilakukan yaitu melakukan proses *update* bobot. Pada proses *update* bobot menggunakan fungsi *Gaussian,* semua neuron-neuron yang ada pada jaringan SOM dilakukan proses *update*. Namun, neuron pemenang dan neuron yang semakin dekat dengan neuron pemenang memiliki nilai *update* bobot yang jauh lebih besar dibandingkan dengan neuron yang jauh dengan neuron pemenang.

Nilai perubahan *update* bobot untuk masing-masing neuron dapat dihitung sebagai berikut.

$$\Delta w_{ij}(t) = h_{ij}^{c}(x_{ij} - w_{ij}(old))$$
$$= \alpha(t) \times e^{\left(\frac{-\|r_{ij} - r_c\|^2}{2 \times \eta(t)^2}\right)} (x_{ij} - w_{ij}(old))$$

$\|r_{ij} - r_c\|$ adalah jarak antara neuron yang akan di-*update* dengan neuron pemenang. Jarak ini dapat dihitung dengan jarak Euclidian. Sebagai contoh, proses perhitungan jarak pada neuron 1,1 atribut ke 1 adalah sebagai berikut:

$$\|r_{ij} - r_c\| = \sqrt{((r_i - r_{ic})^2 + (r_j - r_{jc})^2)}$$
$$= \sqrt{((1-2)^2 + (1-2)^2)}$$
$$= \sqrt{2}$$
$$= 1.4142136$$
$$\|r_{ij} - r_c\|^2 = 2$$

Nilai perubahan *update* bobot pada neuron 1,1 atribut ke-1 adalah sebagai berikut:

$$\Delta w_{ij}(t) = \alpha(t) \times e^{\left(\frac{-\|r_{ij} - r_c\|^2}{2 \times \eta(t)^2}\right)} (x_{ij} - w_{ij}(old))$$
$$= 0.5 \times e^{\left(\frac{-1 \times 2}{2 \times 0.35826^2}\right)} (0.0310458 - 0.8940559)$$
$$= 0.5 \times e^{(-7.79118)} (-0.8630101)$$
$$= 0.5 \times 0.0413364 \times -0.8630101$$
$$= -0.0001784$$

Sehingga, bobot baru pada neuron 1,1 atribut ke-1 untuk data *input* Kabupaten Pacitan adalah sebagai berikut:

$$w_{ij}(t) = w_{ij}(old) + \Delta w_{ij}(t)$$
$$= 0.894056 + (-0.0001784)$$
$$= 0.8938775$$

Sebagai contoh selanjutnya, perhitungan untuk *update* bobot pada neuron 2,1 atribut ke-2 dan neuron 2,2 atribut ke-3 adalah sebagai berikut.

- *Update* bobot neuron 2,1 atribut ke-2

$$\Delta w_{ij}(t) = \alpha(t) \times e^{\left(\frac{-\|r_{ij} - r_c\|^2}{2 \times \eta(t)^2}\right)} (x_{ij} - w_{ij}(old))$$
$$= 0.5 \times e^{\left(\frac{-1 \times 1^2}{2 \times 0.35826^2}\right)} (0.0996 - 0.4458)$$
$$= 0.5 \times e^{(-3.89559)} (-0.3462)$$
$$= 0.5 \times 0.020331 \times -0.3462$$
$$= 0.003519$$

$$w_{ij}(t) = w_{ij}(old) + \Delta w_{ij}(t)$$
$$= 0.4458 + (-0.003519)$$
$$= 0.44291.$$

- *Update* bobot neuron 2,2 atribut ke-3

$$\begin{aligned}\Delta w_{ij}(t) &= \alpha(t)\times e^{\left(\frac{-\|r_{ij}-r_c\|^2}{2\times\eta(t)^2}\right)}(x_{ij}-w_{ij}(old))\\ &= 0.5\times e^{\left(\frac{-1\times 0^2}{2\times 0.35826^2}\right)}(0.-0.)\\ &= 0.5\times e^{(-0.)}(-0.)\\ &= 0.5\times 0.\times -0.\\ &= 0.\end{aligned}$$

$$\begin{aligned}w_{ij}(t) &= w_{ij}(old)+\Delta w_{ij}(t)\\ &= 0.14938+0.09348\\ &= 0.24286\end{aligned}$$

Pada data *input* selanjutnya, yaitu data *input* Kabupaten Jember, proses *update* bobot menggunakan nilai bobot hasil dari *update* bobot pada data *input* sebelumnya yaitu data *input* Kabupaten Pacitan. Sebagai contoh, proses *update* pada neuron 1,1 atribut 1 untuk data *input* Kabupaten Jember adalah sebagai berikut:

$$\begin{aligned}\Delta w_{ij}(t) &= \alpha(t)\times e^{\left(\frac{-\|r_{ij}-r_c\|^2}{2\times\eta(t)^2}\right)}(x_{ij}-w_{ij}(old))\\ &= 0.5\times e^{\left(\frac{-1\times 0^2}{2\times 0.35826^2}\right)}(0.-0.)\\ &= 0.5\times e^{(0)}(-0.)\\ &= 0.5\times 1\times -0.\\ &= 0.\end{aligned}$$

$$\begin{aligned}w_{ij}(t) &= w_{ij}(old)+\Delta w_{ij}(t)\\ &= 0.8938775+(-0.04252875)\\ &= 0.8513505\end{aligned}$$

Nilai bobot jaringan SOM setelah semua data input (Kab. Pacitan, Kab. Jember, Kab. Banyuwangi) diproses untuk 1 kali *epoch*/iterasi dapat dilihat pada Tabel 4.11.

Tabel 4.11 Hasil update bobot untuk epoch pertama

Neuron id	Atribut 1	Atribut 2	Atribut 3
Neuron 1 1	0,87257067815426	0,937856177062769	0,128549789783647
Neuron 1 2	0,186039114143678	0,433992976434553	0,172244845520943
Neuron 2 1	0,439285814933999	0,949565098327245	0,395524537377851

Tabel 4.11 Hasil *update* bobot untuk *epoch* pertama (lanjutan)

Neuron id	Atribut 1	Atribut 2	Atribut 3
Neuron 2 2	0,130298778134774	0,061485059735993	0,242807546003385

Selanjutnya di *epoch* kedua, terjadi perubahan nilai pada konstanta *alpha* dan *eta*. Nilai ini berubah sesuai dengan rumus perubahan konstanta *alpha* dan *beta* yang digunakan.

$$\alpha(t) = \alpha(0).\frac{1}{t}$$

$$\alpha(1) = 0.5.\frac{1}{2}$$

$$= 0.25$$

$$\eta(t) = \eta(0).e^{(\frac{t}{T_{max}})}$$

$$\eta(1) = 0.5.e^{(\frac{2}{3})}$$

$$= 0.256708$$

Langkah selanjutnya sama seperti pada *epoch* ke-1, yaitu penentuan neuron pemenang dan perubahan atau *update* bobot. Setiap *epoch* dan data *input* yang diproses menggunakan bobot dari hasil proses *update* bobot pada tahap sebelumnya. Proses ini diulangi sebanyak data *input* yang diproses dan sebanyak jumlah *epoch* maksimal yang ditentukan pengguna. Sebagai contoh, jika terdapat sebanyak 3 *epoch* dan 3 data *input* pada proses *training*, maka setidaknya terdapat 9 proses *update* bobot yang terjadi. Selain itu pada setiap *epoch*-nya, nilai *alpha* dan *eta* berubah sesuai dengan rumus yang digunakan.

Hasil bobot jaringan SOM setelah melewati 3 *epoch* dapat dilihat pada Tabel 4.12.

Tabel 4.12 Hasil proses *training* setelah diulangi sebanyak 3 kali *epoch*

Neuron id	Atribut 1	Atribut 2	Atribut 3
Neuron 1 1	0,862637229201454	0,974244791170563	0,107189803343149
Neuron 1 2	0,186188133873747	0,453589389153651	0,162426146151518
Neuron 2 1	0,439338533837196	0,949469192071223	0,395441975041957
Neuron 2 2	0,093078973023611	0,075784583194603	0,277883714109681

4.2.5 Penentuan *cluster* atau *clustering*

Proses penentuan *cluster* atau *clustering* pada dasarnya adalah proses untuk menentukan neuron pemenang pada masing-masing data input dengan menggunakan bobot jaringan SOM yang telah dilakukan proses *training.*

Proses penentuan neuron pemenang pada data *input* Pacitan adalah sebagai berikut:

$$\begin{aligned} d(pacitan, neuron_{11}) &= \sqrt{((x_1 - w_{11.1})^2 + (x_2 - w_{11.2})^2 + (x_3 - w_{11.3})^2)} \\ &= \sqrt{((0.031 - 0.862)^2 + (0.099 - 0.974)^2 + (0.336 - 0.107)^2)} \\ &= \sqrt{(0.6905 + 0.7656 + 0.0524)} \\ &= 1.22821008 \end{aligned}$$

$$\begin{aligned} d(pacitan, neuron_{12}) &= 0.42380 \\ d(pacitan, neuron_{21}) &= 0.94469 \\ d(pacitan, neuron_{22}) &= 0.08850 \end{aligned}$$

$$\begin{aligned} c_{pacitan} &= \underset{i,j}{argmin}(\{1.22821, 0.42380, 0.94469, 0.08850\}) \\ &= [2,2] \end{aligned}$$

Proses penentuan neuron pemenang pada data *input* Jember adalah sebagai berikut:

$$\begin{aligned} c_{jember} &= \underset{i,j}{argmin}(\{0.07086, 0.82346, 0.44586, 1.16996\}) \\ &= [1,1] \end{aligned}$$

Proses penentuan neuron pemenang pada data *input* Banyuwangi adalah sebagai berikut:

$$\begin{aligned} c_{banyuwangi} &= \underset{i,j}{argmin}(\{0.07059, 0.90114, 0.57348, 1.244008\}) \\ &= [1,1] \end{aligned}$$

4.2.6 Perhitungan *Silhouette Coefficient*

Perhitungan Silhouette Coefficient dilakukan untuk mengetahui kualitas dari *cluster* yang terbentuk. Perhitungan Silhouette Coefficient memperhitungkan setiap objek data di dalam sebuah *cluster* yang terbentuk. Karena memperhitungkan jarak antara objek data dengan data yang lain. Perhitungan Silhouette Coefficient membutuhkan data dalam bentuk normalisasi untuk perhitungan jaraknya.

Sedangkan, bentuk *cluster* yang akan digunakan sebagai contoh perhitungan manual Silhouette Coefficient dapat dilihat pada Tabel 4.13 dan Tabel 4.14.

- Cluster ke-1:

Tabel 4.13 Contoh *cluster* terbentuk untuk perhitungan manual *Silhouette Coefficient* (1)

Neuron	2	2	
Pacitan	0,031045751633987	0,99616858237548	0,336344041876219

- Cluster ke-2:

Tabel 4.14 Contoh *cluster* terbentuk untuk perhitungan manual *Silhouette Coefficient* (2)

Neuron	1	1	
Jember	0,808823529411765	0,992337164750958	0,149594580724623
Banyuwangi	0,893790849673203	1	0,049317458688289

Pada kasus ini, masing-masing ketiga data tersebut akan dihitung nilai *Silhouette Coefficient*-nya.

- Perhitungan *Silhouette Coefficient* data Pacitan

  Karena data Pacitan adalah masuk ke dalam sebuah cluster yang hanya memiliki satu anggota (yaitu data Pacitan itu sendiri maka nilai Silhouette Coefficient-nya adalah 0.

  $$s(pacitan)=0$$

- Perhitungan *Silhouette Coefficient* data Jember

  1. Menghitung rata-rata jarak objek data Jember ke objek data *cluster*-nya (*cluster* ke 2)

$$\begin{aligned} d(jember, banyuwangi) &= \sqrt{\sum_{j=1}^{j=3} (x_{jember,\,j} - x_{banyuwangi,\,j})^2} \\ &= \sqrt{((0.808-0.893)^2+(0.992-1)^2+(0.149-0.049)^2)} \\ &= \sqrt{(0.007+0.0000587+0.01005)} \\ &= 0.1316 \end{aligned}$$

$$\begin{aligned} a(jember) &= \frac{1}{(|C_2|-1)} d(jember, banyuwangi) \\ &= \frac{1}{(2-1)} \times 0.1316 \\ &= 0.1316 \end{aligned}$$

  2. Mencari jarak terkecil dari masing-masing *cluster* yang terbentuk. Karena pada kasus ini *cluster* yang terbentuk hanya 2, maka jarak *cluster* terkecil adalah *cluster* ke-1.

$$d(jember, pacitan) = \sqrt{\sum_{j=1}^{j=3} (x_{jember,j} - x_{pacitan,j})^2}$$
$$= \sqrt{((0.808-0.031)^2+(0.992-0.099)^2+(0.149-0.336)^2)}$$
$$= \sqrt{(0.604+0.796+0.034)}$$
$$= 1.198$$

$$d(jember, C_1) = \frac{1}{(|C_1|)} d(jember, pacitan)$$
$$= \frac{1}{(1)} \times 1.198$$
$$= 1.198$$

$$b(jember) = min\{d(jember, C_1)\}$$
$$= 1.198$$

3. Menghitung nilai *Silhouette Coefficient* data Jember

$$s(jember) = \frac{b(jember) - a(jember)}{max\{a(jember), b(jember)\}}$$
$$= \frac{1.198-0.131}{max\{0.131, 1.198\}}$$
$$= \frac{1.066}{1.198}$$
$$= 0.8901$$

- Perhitungan *Silhouette Coefficient* data Banyuwangi

1. Menghitung rata-rata jarak objek data Banyuwangi ke objek data *cluster*-nya (*cluster* ke 2).

$$d(jember, banyuwangi) = \sqrt{\sum_{j=1}^{j=3} (x_{jember,j} - x_{banyuwangi,j})^2}$$
$$= \sqrt{((0.808-0.893)^2+(0.992-1)^2+(0.149-0.049)^2)}$$
$$= \sqrt{(0.007+0.0000587+0.01005)}$$
$$= 0.1316$$

$$a(banyuwangi) = \frac{1}{(|C_2|-1)} d(banyuwangi, jember)$$
$$= \frac{1}{(2-1)} \times 0.1316$$
$$= 0.1316$$

2. Mencari jarak terkecil dari masing-masing *cluster* yang terbentuk. Karena pada kasus ini *cluster* yang terbentuk hanya 2, maka jarak *cluster* terkecil adalah *cluster* ke-1.

$$d(banyuwangi, pacitan) = \sqrt{\sum_{j=1}^{j=3} (x_{banyuwangi,j} - x_{pacitan,j})^2}$$
$$= \sqrt{((0.893-0.031)^2+(1-0.099)^2+(0.049-0.336)^2)}$$
$$= \sqrt{(0.744+0.810+0.082)}$$
$$= 1.2796$$

$$d(banyuwangi, C_1) = \frac{1}{(|C_1|)} d(banyuwangi, pacitan)$$
$$= \frac{1}{(1)} \times 1.2796$$
$$= 1.2796$$

$$b(banyuwangi) = min\{d(banyuwangi, C_1)\}$$
$$= 1.2796$$

3. Menghitung nilai Silhouette Coefficient data Banyuwangi

$$s(banyuwangi) = \frac{b(banyuwangi) - a(banyuwangi)}{max\{a(banyuwangi), b(banyuwangi)\}}$$
$$= \frac{1.2796 - 0.131}{max\{0.131, 1.2796\}}$$
$$= \frac{1.147}{1.2796}$$
$$= 0.8971$$

Rata rata global dari *Silhouette Coefficient* untuk semua *dataset* adalah dapat dihitung sebagai berikut:

$$avg\ \ s(i) = \frac{1}{N} \times (s(pacitan) + s(jember) + s(banyuwangi))$$
$$= \frac{1}{3} \times (0 + 0.89016 + 0.89711)$$
$$= \frac{1}{3} \times 1.7872$$
$$= 0.5957$$

4.3 Perancangan Pengujian

Untuk menguji apakah algoritme dapat berjalan dengan baik dan sesuai dengan tujuan penelitian, maka perlu dilakukan sebuah pengujian. Sebelum melakukan pengujian, sebuah pengujian harus dirancang terlebih dahulu. Pengujian di sini mencakup pengujian beberapa parameter algoritme yang ada di baik algoritme *Self-Organizing Maps* (SOM) maupun parameter yang ada di algoritme K-*Nearest Neighbors* (KNN).

4.3.1 Pengujian parameter pada algoritme *Self-Organizing Maps* (SOM)

Pada algoritme SOM, terdapat beberapa parameter nilai yang harus diujikan. Parameter nilai tersebut salah satunya adalah *learning rate* atau konstanta

alpha. Parameter nilai lain yang diujikan yaitu nilai signifikansi *update* bobot neuron tetangga atau nilai *eta.* Kemudian, pengujian *epoch* dilakukan untuk mengetahui jumlah *epoch* yang optimal. Sedangkan pengujian jumlah neuron dilakukan untuk mengetahui bentuk *cluster* terbaik yang dihasilkan oleh jaringan SOM.

4.3.1.1 Pengujian konstanta alpha pada algoritme SOM

Pengujian ini mengambil nilai *learning rate* yang memberikan hasil terbaik ketika diukur menggunakan *Silhouette Coefficient*. Dalam hal ini, hasil terbaik pengujian yaitu ketika pengujian memberikan nilai rata-rata global *Silhouette Coefficient* tertinggi dibandingkan nilai parameter lainnya yang diujikan. Nilai *alpha* yang diujikan yaitu 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, dan 0.9. Proses pengujian dilakukan sebanyak 10 kali untuk setiap nilai *alpha* dan diambil rata-rata dari 10 percobaan tersebut.

Variabel kontrol yang akan digunakan pada pengujian ini berupa nilai eta, jumlah *epoch*, jumlah neuron dan nilai K dengan nilai sebagai berikut:

1. Nilai eta: 0.5
2. Jumlah epoch: 200
3. Jumlah neuron: 5x5
4. Nilai K pada KNN: 0

Skenario pengujian tersebut dapat dilihat pada Tabel 4.15.

Tabel 4.15 Perancangan skenario pengujian nilai konstanta *alpha* pada algoritme SOM

No.	Parameter nilai *alpha* jaringan SOM	Percobaan ke-	Nilai *average global Silhouette Coefficient*	Rata-rata	Nilai tertinggi
1	Nilai *alpha* ke-1	1			
		2			
		...	...		
		10			
2	Nilai *alpha* ke-2	1			
		2			
		...	...		
		10			
...	...	...	...	...	

Tabel 4.15 Perancangan skenario pengujian nilai konstanta alpha pada algoritme SOM (lanjutan)

No.	Parameter nilai *alpha* jaringan SOM	Percobaan ke-	Nilai *average global Silhouette Coefficient*	Rata-rata	Nilai tertinggi
n	Nilai *alpha* ke-n	1			
		2			
		...			
		10			

4.3.1.2 Pengujian konstanta *eta* pada algoritme SOM

Konstanta *eta* merupakan konstanta yang menentukan signifikasi proses *update* bobot yang terjadi pada neuron tetangga. Nilai *eta* ini digunakan ketika menggunakan jaringan SOM dengan fungsi *Gaussian* sebagai perhitungan *update* bobotnya. Skenario pengujiannya juga sama dengan pengujian terhadap nilai *alpha,* yaitu menguji beberapa nilai *eta* dan mengambil nilai eta yang memberikan hasil terbaik. Nilai *eta* yang diujikan yaitu 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, dan 0.9. Pengujian ini diukur menggunakan nilai *Silhouette Coefficient* dan mengambil nilai parameter yang memberikan nilai rata-rata global *Silhouette Coefficient* terbesar. Pengujian dilakukan sebanyak 10 untuk masing-masing nilai *eta* dan diambil nilai rata-rata *Silhouette Coefficient* dari 10 kali percobaan tersebut.

Variabel kontrol yang akan digunakan pada pengujian ini berupa nilai *alpha*, jumlah *epoch*, jumlah neuron dan nilai K dengan nilai sebagai berikut:

1. Nilai *alpha*: 0.5
2. Jumlah *epoch*: 200
3. Jumlah neuron: 5x5
4. Nilai K pada KNN: 0

Skenario pengujian tersebut dapat dilihat pada Tabel 4.16.

Tabel 4.16 Perancangan skenario pengujian nilai konstanta *eta* pada algoritme SOM

No.	Parameter nilai *eta* jaringan SOM	Percobaan ke-	Nilai *average global Silhouette Coefficient*	Rata-rata	Nilai tertinggi
1	Nilai *eta* ke-1	1			

Tabel 4.16 Perancangan skenario pengujian nilai konstanta eta pada algoritme SOM (lanjutan)

No.	Parameter nilai *eta* jaringan SOM	Percobaan ke-	Nilai *average global Silhouette Coefficient*	Rata-rata	Nilai tertinggi
		2			
		...			
		10			
2	Nilai *eta* ke-2	1			
		2			
		...			
		10			
...	...	...	...	...	
n	Nilai *eta* ke-n	1			
		2			
		...			
		10			

4.3.1.3 Pengujian jumlah *epoch* pada algoritme SOM

Skenario pengujian *epoch* hampir sama dengan skenario pengujian nilai *alpha* dan *eta.* Pengujian dilakukan sebanyak 10 kali dan diambil nilai rata-rata *Silhouette Coefficient* untuk setiap jumlah *epoch* yang diujikan. Jumlah *epoch* yang diujikan yaitu 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 dan 200. Pengujian jumlah epoch dilakukan untuk mencari jumlah *epoch* yang memberikan nilai rata-rata global *Silhouette Coefficient* yang tertinggi.

Variabel kontrol yang akan digunakan pada pengujian ini berupa nilai *alpha*, nilai *eta*, jumlah neuron dan nilai K dengan nilai sebagai berikut:

1. Nilai *alpha*: 0.5
2. Nilai *eta*: 0.5
3. Jumlah neuron: 5x5
4. Nilai K pada KNN: 0

Skenario pengujian jumlah *epoch* dapat dilihat pada Tabel 4.17

Tabel 4.17 Perancangan skenario pengujian jumlah *epoch* pada algoritme SOM

No.	Parameter jumlah *epoch* jaringan SOM	Percobaan ke-	Nilai *average global Silhouette Coefficient*	Rata-rata	Nilai tertinggi
1	Jumlah *epoch* ke-1	1			
		2			
		...	...		
		10			
2	Jumlah *epoch* ke-2	1			
		2			
		...	...		
		10			
...	...	...	...	...	
n	Jumlah *epoch* ke-n	1			
		2			
		...	...		
		10	...		

4.3.1.4 Pengujian jumlah neuron pada algoritme SOM

Pengujian ini digunakan untuk mengetahui jumlah neuron yang cocok ketika algoritme SOM dijalankan pada *dataset* PMKS. Pengujian ini menggunakan nilai *Silhouette Coefficient* sebagai tolak ukur pengujian. Pengujian ini dilakukan sebanyak 10 kali dan diambil rata-rata *Silhouette Coefficient* untuk setiap jumlah neuron yang diujikan. Karena jenis arsitektur SOM yang digunakan adalah arsitektur 2D-Rectangular, maka jumlah neuron yang diujikan mengikuti deret geometri. Dalam hal ini jumlah neuron yang diujikan yaitu 2x2, 3x3, 4x4, 5x5 dan 6x6. Pengujian jumlah neuron ini menggunakan jumlah *epoch* serta nilai *alpha* dan eta uang optimal dari hasil pengujian sebelumnya.

Variabel kontrol yang akan digunakan pada pengujian ini berupa nilai *alpha*, nilai eta, jumlah *epoch* dan nilai K dengan nilai sebagai berikut:

1. Nilai *alpha*: 0.5
2. Nilai *eta*: 0.5
3. Jumlah *epoch*: 200

4. Nilai K pada KNN: 0

Skenario pengujian jumlah neuron dapat dilihat pada Tabel 4.18.

Tabel 4.18 Perancangan skenario pengujian jumlah neuron pada algoritme SOM

No.	Jumlah neuron	Percobaan ke-	Nilai *average global Silhouette Coefficient*	Rata-rata	Nilai tertinggi
1	2x2	1			
		2			
		...	...		
		10			
2	3x3	1			
		2			
		...	...		
		10			
3	....	1			
		2			
		...			
		10			
4	6x6	1			
		2			
		...			
		10			

4.3.2 Pengujian nilai K pada algoritme *K-Nearest Neighbors*

Pengujian ini dilakukan untuk mengetahui berapa jumlah K nilai tetangga terdekat yang akan digunakan untuk mengisi data yang kosong. Pengujian ini dilakukan sebanyak 10 kali dan menggunakan tolak ukur berupa nilai *Silhouette Coefficient*. Pengujian ini menggunakan nilai parameter konstanta *alpha/eta*, jumlah *epoch* dan jumlah neuron dari SOM yang optimal pada pengujian sebelumnya. Pengujian ini juga dilakukan dengan membandingkan antara ketika tidak menggunakan pengisian data kosong dan menggunakan pengisian data

kosong. Sehingga hasil pengujian ketika skenarionya tidak menggunakan pengisian data kosong dapat mengambil dari pengujian jumlah SOM yang memberikan hasil optimal sebelumnya.

Variabel kontrol yang akan digunakan pada pengujian ini berupa nilai alpha, nilai eta, jumlah epoch dan nilai K dengan nilai sebagai berikut:

1. Nilai alpha: 0.5
2. Nilai eta: 0.5
3. Jumlah epoch: 200
4. Jumlah neuron: 5x5

Skenario pengujian pengaruh pengisian data kosong dapat dilihat pada Tabel 4.19.

Tabel 4.19 Perancangan skenario pengujian pengaruh *missing-value imputation* terhadap proses clustering

No.	Nilai K	Percobaan ke-	Nilai *average global Silhouette Coefficient*	Rata-rata	Nilai tertinggi
1	Tidak menggunakan *missing-value imputation* (K = 0)	1			
		2			
		...	...		
		10			
2	K = 1	1			
		2			
		...	...		
		10			
...	...	...	...	...	
6	K = 6	1			
		2			
		...	...		
		10			

4.4 Perancangan Antarmuka

Antarmuka dalam terminologi komputer merupakan sebuah penghubung antara pengguna (*user*) dengan sistem atau program yang ada di dalam komputer sehingga sistem dan program tersebut dapat digunakan sesuai dengan fungsionalitas yang telah ditentukan sebelumnya. Pada penelitian ini, hal yang dapat dilakukan dari *user* sebelum dapat menggunakan program atau algoritme untuk yaitu dengan memasukkan *input* data dan parameter-parameter nilai dari algoritme yang akan diujikan. Kemudian terdapat tampilan terkait proses *clustering* yang sedang berjalan beserta hasil dari *cluster* yang terbentuk serta nilai atau skor dari pengujian tersebut.

4.4.1 Antarmuka Mulai *Clustering*

Pada antarmuka mulai *clustering* seperti yang ditunjukkan pada Gambar 4.10, pengguna akan memasukkan sebuah *file* terlebih dahulu yang merupakan dataset beserta dengan parameter-parameter algoritme yang akan diujikan. *Dataset* yang dapat digunakan dan dimasukkan ke dalam program yaitu *dataset* dengan format **.csv* dengan struktur data matriks tabel antara atribut dengan baris data. Format **.csv* yang dapat dimasukkan ke sistem yaitu *file .csv* yang memiliki *header* pada masing-masing atributnya. Setelah pengguna telah memasukkan *dataset* yang akan diujikan, Kemudian pengguna akan mengisi nilai beberapa parameter algoritme.

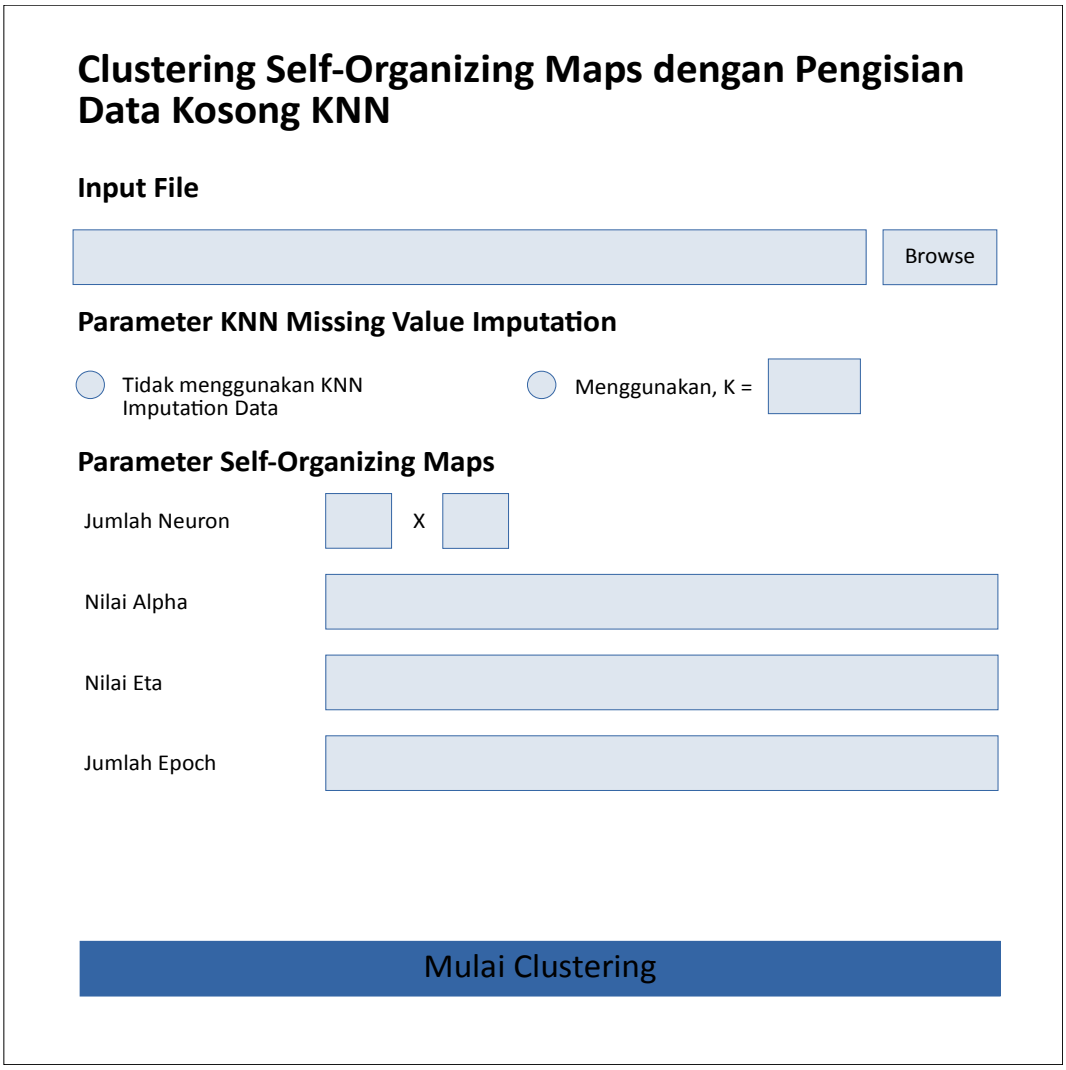
Pada parameter algoritme pengisian *missing value* menggunakan KNN, pengguna akan memilih opsi antara tidak menggunakan pengisian data kosong atau memilih menggunakan opsi untuk melakukan pengisian nilai kosong dengan memasukkan parameter K yang akan digunakan untuk proses pengisian nilai itu sendiri.

Pada parameter dari algoritme SOM, pengguna akan mengisi beberapa nilai yang akan digunakan dalam proses *training* menggunakan algoritme SOM. Parameter tersebut antara lain yaitu jumlah neuron, jumlah *epoch*, nilai konstanta *alpha* dan nilai konstanta *eta*. Karena menggunakan arsitektur 2D, maka jumlah neuron yang dimasukkan ke dalam program yaitu direpresentasikan dalam bentuk panjang neuron dikalikan dengan lebar neuron.

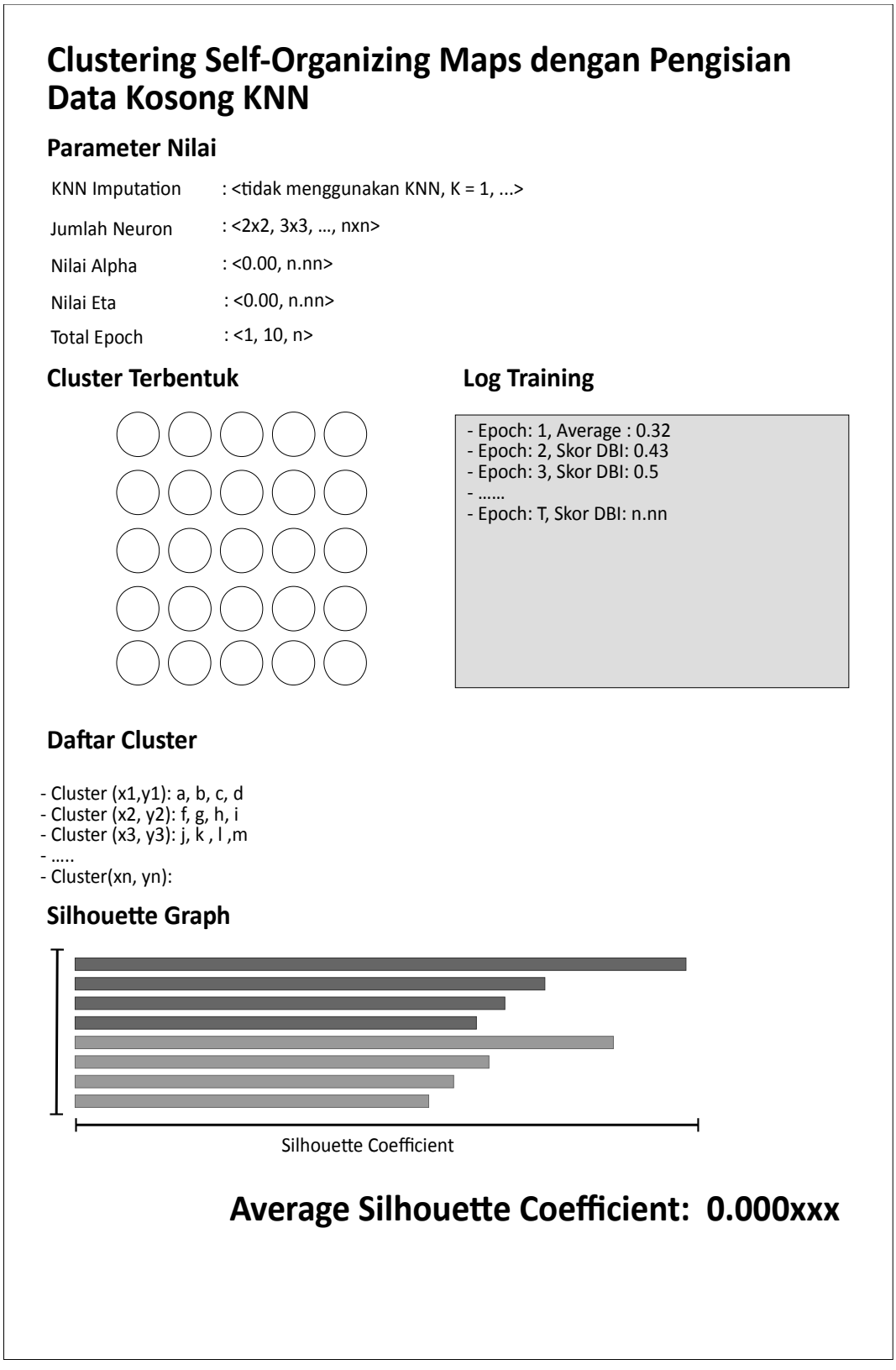
4.4.2 Antarmuka Hasil *Clustering*

Pada antarmuka hasil *clustering* seperti yang ditunjukkan oleh Gambar 4.11, pengguna akan mendapatkan tampilan berupa parameter yang telah digunakan untuk proses *clustering*. Kemudian, pengguna juga mendapatkan tampilan berupa log dalam format *epoch* yang sedang berlangsung beserta dengan skornya untuk tiap-tiap *epoch* yang berlangsung tersebut. Kemudian pengguna juga mendapatkan tampilan berupa bentuk dari *cluster* yang telah terbentuk beserta daftar dari anggota dari masing-masing *cluster* yang terbentuk. Setelah proses *clustering* selesai atau telah mencapai batas *epoch* yang ditentukan,

pengguna akan mendapatkan tampilan berupa grafik *Silhouette Coefficien*t dan nilai rata-rata *Silhouette Coefficient* dari pengujian yang telah dilakukan tersebut.



Gambar 4.10 Perancangan antarmuka mulai clustering



Gambar 4.11 Perancangan antarmuka hasil clustering

BAB 5 IMPLEMENTASI

5.1 Implementasi Algoritme

Bagian ini menjelaskan tentang implementasi algoritme yang sudah dirancang pada bab sebelumnya. Bahasa yang digunakan pada implementasi algoritme adalah bahasa pemrograman Python. Selanjutnya, tahap implementasi algoritme dibagi menjadi 2 bagian, yaitu implementasi algoritme *K-Nearest Neighbors* untuk pengisian data kosong dan implementasi algoritme *Self-Organizing Maps* untuk *clustering*.

5.1.1 Implementasi kode program Pengisian *missing-value* menggunakan *K-Nearest Neighbors*

Implementasi pengisian *missing-value* menggunakan algoritme KNN terdiri dari 3 fungsi. Terdapat satu fungsi utama yaitu impute_dataset yang digunakan untuk mengisi dataset yang memiliki nilai kosong. Dua fungsi selanjutnya, yaitu fungsi get_dist_pair dan get_euclidian_dist, digunakan untuk mendapatkan daftar data jarak pada antara keseluruhan data dengan data lainnya.

5.1.1.1 Implementasi fungsi mendapatkan jarak Euclidian antar dua data

Pada algoritme KNN, jarak antar dua data digunakan untuk menentukan data mana yang memiliki jarak terdekat. Pada implementasinya, proses perhitungan jarak memiliki parameter *input* berupa *dataset* yang digunakan serta 2 indeks dari data pada *dataset input* yang akan dihitung jarak datanya. Implementasi tersebut dapat dilihat pada Kode Sumber 5.1.

Algoritme 1: Fungsi mendapatkan jarak euclidian antar 2 data

```
1   def get_euclidian_dist(input_dataset, a, b):
2       inverted_input_data = list(map(list, zip(*input_dataset)))
3       sum_sq = 0
4       for id_attr in range(0, len(input_dataset[0])):
5           attr_df = df(inverted_input_data[id_attr])
6
7           max_attr = attr_df.max(skipna=True)
8           min_attr = attr_df.min(skipna=True)
9           val_a = input_dataset[a][id_attr]
10          val_b = input_dataset[b][id_attr]
11          sq_dist = 1 if (isnan(val_a) or isnan(val_b))
                        else ((abs(val_b - val_a)
                                / (max_attr - min_attr)) ** 2)
12          sum_sq += sq_dist
13      euc_dist = sqrt(sum_sq)
14      return (a, b, euc_dist)
15
```

Kode Sumber 5.1 Mendapatkan jarak Euclidian antar dua data

Penjelasan dari Kode Sumber 5.1 adalah sebagai berikut:

Baris 1 : Melakukan inversi atau pembalikan *list* dengan atribut

sebagai baris dan data sebagai kolom. Sehingga dapat perhitungan diproses dengan mudah pada iterasi.

Baris 4-10 : Iterasi untuk setiap atribut data. Setiap atribut data dicari nilai maksimal dan minimalnya. Kemudian, nilai data ke a dan b untuk atribut yang terpilih diinisialisasi ke dalam variabel val_a dan val_b.

Baris 11 : Jika salah satu nilai pada data ke a ataupun b nilainya kosong, maka *square distance* akan di-*set* ke 1. Namun jika tidak, maka nilainya dihitung menggunakan rumus *normalized squared distance.*

Baris 12-13 : Masing-masing nilai *square* untuk setiap *distance* akan dijumlahkan. *Euclidian distance* dari dua data a dan b yaitu hasil akar dari penjumlahan *normalized squared distance.*

Baris 14 : Nilai yang dikembalikan yaitu berupa *tuple* indeks data ke a dan b beserta nilai jarak *Euclidian distance.*

5.1.1.2 Implementasi fungsi mendapatkan pasangan jarak

Fungsi ini akan mengembalikan nilai berupa daftar semua kemungkinan pasangan data dengan data lain beserta nilai jarak *Euclidian*-nya. *Input* dari fungsi ini adalah sebuah *dataset*. Implementasi tersebut dapat dilihat pada Kode Sumber 5.2.

Algoritme 2: Fungsi mendapatkan daftar jarak Euclidan pada pasangan data

```
1   def get_dist_pair(input_dataset):
2       pair_dist_list = list()
3       for i in range(0, len(input_dataset)):
4           for j in range(0, len(input_dataset)):
5           if ((i != j)
                and ((i, j) not in pair_dist_list)
                and ((j, i) not in pair_dist_list)):
6                       pair_dist_list.append((i, j))
7
8       with Pool() as pool:
9           dist = pool.starmap(get_euclidian_dist,
                                [(input_dataset, a, b)
                                    for a, b in pair_dist_list])
10          pool.close()
11          pool.join()
12      return dist
13
```

Kode Sumber 5.2 Mendapatkan daftar jarak pada pasangan data

Penjelasan dari Kode Sumber 5.2 adalah sebagai berikut:

Baris 2-6 : Membuat *list* yang berisi semua kombinasi pasangan yang terjadi antara data satu dengan data lainnya pada *dataset input*. Kombinasi tersebut disimpan sementara dalam bentuk *list of tuple.*

Baris 7-11 : Dari *list* kombinasi tersebut, masing-masing setiap pasangan dihitung jarak Euclidian *distance*-nya dan disimpan di *list of tuple* dengan format (a, b, nilai_euclidian_distance). Penggunaan pool.starmap digunakan untuk mempercepat proses perhitungan.

Baris 12 : Mengembalikan data *list* kombinasi pasangan data beserta jarak *Euclidian*-nya.

5.1.1.3 Implementasi fungsi *impute dataset*

Fungsi impute_dataset digunakan untuk mengisi dataset yang memiliki nilai kosong. Parameter yang menjadi inputan fungsi tersebut yaitu *dataset* dan nilai K. *Output* dari fungsi ini adalah sebuah *dataset* dengan struktur yang sama namun nilai kosongnya telah terisi. Jika pengguna memasukkan K bernilai 0, artinya pengguna tidak menggunakan KNN untuk mengisi data kosong. Data kosong yang ada pada *dataset* akan digantikan dengan nilai 0. Implementasi tersebut dapat dilihat pada Kode Sumber 5.3.

Algoritme 3: Fungsi mengisi dataset yang memiliki nilai kosong

```
1   def impute_dataset(input_dataset, K):
2       imputed_dataset = list()
3       pair_dist_list = get_dist_pair(input_dataset)
4       for row in range(0, len(input_dataset)):
5           filtered_pair_dist_list = [(x, y, dist) if x == row
                        else (y, x, dist)
                          for (x, y, dist) in pair_dist_list
                            if x == row or y == row ]
6           sorted_dist = sorted(filtered_pair_dist_list,
                             key=lambda x: x[2])
7           imputed_dataset_attr = list()
8           for attr in range(0, len(input_dataset[row])):
9               new_val = input_dataset[row][attr]
10              if isnan(input_dataset[row][attr]):
11                  if K == 0:
12                      imputed_dataset_attr.append(0)
13                      continue
14
15                  sum_imp = 0
16                  count = 0
17                  for a, b, dist in sorted_dist:
18                      # wk = 1 / (dist ** 2)
19                      if isnan(input_dataset[b][attr]):
20                          continue
21                      sum_imp += (input_dataset[b][attr])
22                      count += 1
23                      if count == K:
24                          break
25                  new_val = sum_imp / K
26              imputed_dataset_attr.append(new_val)
27          imputed_dataset.append(imputed_dataset_attr)
28
29      return imputed_dataset
30
```

Kode Sumber 5.3 Implementasi fungsi *impute dataset*

Penjelasan dari Kode Sumber 5.3 adalah sebagai berikut:

Baris 3 : Memanggil fungsi get_pair_dist yang merupakan fungsi untuk mendapatkan *list* semua kombinasi pasangan data pada dataset_input beserta data jaraknya.

Baris 4-6 : Melakukan iterasi untuk setiap baris data pada *dataset*. Kemudian melakukan *filtering* bahwa data jarak yang diambil adalah data baris yang terpilih dengan data lainnya. Kemudian, hasil *filtering* tersebut diurutkan berdasarkan jarak terpendek.

Baris 8-13 : Melakukan perulangan untuk setiap atribut data. Atribut data yang diproses adalah atribut data yang memiliki nilai kosong pada baris data yang sedang aktif. Jika parameter *input* K bernilai 0, maka data kosong tersebut akan langsung diisi dengan nilai 0.

Baris 15-25 : Melakukan iterasi sebanyak jumlah *list* data jarak baris data yang sedang terpilih dengan data lainnya. Jika pada elemen pertama, data tersebut bernilai kosong, maka iterasi dilanjutkan ke data selanjutnya. Jika data lain yang terpilih nilainya tidak kosong, maka data tersebut akan digunakan sebagai kandidat pengisi nilai. Perulangan akan berhenti ketika jumlah K data telah terpenuhi. Kemudian, nilai yang digunakan untuk mengisi data kosong adalah rata-rata dari kandidat pengisi nilai yang sudah terpilih.

Baris 26-29 : Melakukan pengisian pada *dataset* dan *output* dari fungsi ini adalah *dataset* yang sudah diisi nilai kosongnya.

5.1.2 Implementasi kode program algortime Self-Organizing Maps (SOM)

Kode program dari algoritme SOM dibagi menjadi beberapa fungsi. Fungsi-fungsi tersebut antara lain fungsi normalize_data, init_som_net, training dan penentuan_cluster dan dijalankan secara berurutan ketika digunakan.

5.1.2.1 Implementasi fungsi normalisasi data *min-max*

Pada fungsi normalize_data, parameter yang menjadi masukan ke fungsi yaitu sebuah *dataset*. Struktur data dari *dataset input* yaitu berbentuk *multi-dimensional list*. Output dari fungsi ini adalah sebuah *dataset* dengan struktur yang sama dengan *dataset input*, namun telah dinormalisasi nilai-nilainya. Implementasi tersebut dapat dilihat pada Kode Sumber 5.4.

Algoritme 4: Fungsi normalisasi data min-max

```
1  def normalize_data(input_data):
2     # invert input data from data_row * attr to attr * data row
3     normalized_data = list()
4     inverted_input_data = list(map(list, zip(*input_data)))
```

```
5
6       for attr in range(0, len(inverted_input_data)):
7          min_attr = min(inverted_input_data[attr])
8          max_attr = max(inverted_input_data[attr])
9          range_attr = max_attr - min_attr
10         attr_list = list()
11         for row in range(0, len(inverted_input_data[attr])):
12            new_value = (inverted_input_data[attr][row] - min_attr)
                        / (range_attr)
13            attr_list.append(new_value)
14
15         normalized_data.append(attr_list)
16
17      # invert again to original structure of input data
18      normalized_data = list(map(list, zip(*normalized_data)))
19      return normalized_data
20
```

Kode Sumber 5.4 Normalisasi data *min-max*

Penjelasan dari Kode Sumber 5.4 adalah sebagai berikut:

Baris 3-4 : Melakukan inversi atau pembalikan *list* dengan atribut sebagai baris dan data sebagai kolom. Sehingga dapat perhitungan diproses dengan mudah pada iterasi.

Baris 6-9 : Iterasi untuk setiap atribut data. Setiap atribut data dicari nilai maksimal dan minimalnya. Kemudian, nilai data ke a dan b untuk atribut yang terpilih diinisialisasi ke dalam variabel val_a dan val_b.

Baris 8-13 : Melakukan perulangan untuk setiap atribut data. Atribut data yang diproses adalah atribut data yang memiliki nilai kosong pada baris data yang sedang aktif. Jika parameter *input* K bernilai 0, maka data kosong tersebut akan langsung diisi dengan nilai 0.

Baris 11 : Melakukan iterasi pada setiap baris data. Setiap baris data akan dihitung nilai normalisasinya menggunakan properti nilai maksimal dan nilai minimal pada masing-masing atribut.

Baris 12-15 : Melakukan pengisian pada *dataset* ke dalam *list.*

Baris 17-19 : Sebelum *dataset* dikembalikan, *dataset* dilakukan pembalikan dengan struktur *dataset* atribut sebagai kolom dan data sebagai baris.

5.1.2.2 Implementasi fungsi inisialiasi bobot awal jaringan SOM

Pada fungsi init_som_net, parameter *input* yang dibutuhkan oleh fungsi yaitu row_size yang merupakan banyak baris neuron, column_size yang merupakan banyak kolom neuron dan attribute_size yang merupakan jumlah atribut data pada *dataset* yang digunakan. Fungsi ini memiliki *output return value* berupa *list* 3 dimensi yang berisi nilai bilangan desimal *random* antara 0 hingga 1. Implementasi tersebut dapat dilihat pada Kode Sumber 5.5.

Algoritme 5: Fungsi mendapatkan jarak euclidian antar 2 data

```
1   def init_som_net(row_size, column_size, attribute_size):
2      weight = list()
3      for i in range(0, row_size):
4         weight_row = list()
5         for j in range(0, column_size):
6            weight_column = list()
7            for k in range(0, attribute_size):
8               weight_column.append(rd.random())
9            weight_row.append(weight_column)
10        weight.append(weight_row)
11     return weight
12
```

Kode Sumber 5.5 Inisialisasi bobot awal jaringan SOM

Penjelasan dari Kode Sumber 5.5 adalah sebagai berikut:

Baris 3-5	:	Melakukan iterasi sebanyak jumlah baris dan kolom neuron pada jaringan SOM.
Baris 6	:	Melakukan iterasi sebanyak jumlah atribut data.
Baris 8	:	Melakukan pengisian *list* dengan nilai *random* antara 0 hingga 1.
Baris 9-10	:	Masing-masing nilai *random* tersebut ditambahkan ke list 3-dimensi.
Baris 11	:	Nilai *output* fungsi berupa list 3-dimensi yang berisi nilai *random* antara 0 hingga 1.

5.1.2.3 Implementasi fungsi *training* jaringan SOM

Pada fungsi ini, implementasi fungsi dibagi menjadi 3 fungsi. Fungsi yang pertama, yaitu fungsi get_dist_data_from_neuron merupakan fungsi yang digunakan untuk mendapatkan *list* jarak antara 1 data *input* dengan neuron-neuron pada jaringan SOM. Fungsi tersebut nanti akan digunakan pada fungsi one_epoch_training yang merupakan fungsi untuk melakukan *training* jaringan hanya untuk satu kali *epoch*.

Fungsi one_epoch_training dan *training* sendiri memiliki *output return value* yang sama, yaitu *list* 3-dimensi bobot jaringan yang telah di-*training*. Fungsi tersebut juga memiliki parameter *input* yang hampir sama. Masing-masing membutuhkan *input* berupa *dataset input*, data bobot jaringan, nilai *alpha* dan nilai eta. Namun pada fungsi *training*, nilai *alpha* dan *eta* yang digunakan adalah nilai ketika *epoch* masih 1 kali. Sedangkan pada one_epoch_training, nilai *alpha* dan *eta* yang digunakan adalah nilai konstanta yang bergantung pada t atau *epoch*. Selain itu pada fungsi *training*, terdapat parameter max_epoch yang menentukan berapa jumlah *epoch* atau iterasi dilakukan ketika proses *training* berlangsung.

Implementasi fungsi *training* jaringan SOM dapat dilihat pada Kode Sumber 5.6.

Algoritme 6: Fungsi melakukan training jaringan SOM

```
1     def get_dist_data_from_neur(trained_weight, input_data):
2        # find distance from data input to neuron
3        dist_data_to_neur = list()
4        for neur_i in range(0, len(trained_weight)):
5           for neur_j in range(0, len(trained_weight[neur_i])):
6              sumsq = 0
7              for attr in range(0,
                                 len(trained_weight[neur_i][neur_j])):
8                 sumsq += (input_data[attr]
                            - trained_weight[neur_i][neur_j][attr])** 2
9              rootsumsq = sqrt(sumsq)
10             dist_data_to_neur.append(((neur_i, neur_j), rootsumsq))
11       return dist_data_to_neur
12
13
14    def one_epoch_training(dataset_input, trained_weight, alpha_t,
                             eta_t):
15       for row in range(0, len(dataset_input)):
16
17             # find best-matching-unit neuron
18          dist_data_to_neur = get_dist_data_from_neur(
                               trained_weight, dataset_input[row])
19          dist_sorted = sorted(dist_data_to_neur, key=lambda x: x[1])
20          c = dist_sorted[0][0]  # access tuple of neuron index
21
22          # weight update
23          for neur_i in range(0, len(trained_weight)):
24             for neur_j in range(0, len(trained_weight[neur_i])):
25                for attr in range(0,
                                    len(trained_weight[neur_i][neur_j])):
26                   dist_rij_rc = sqrt((neur_i - c[0])**2
                                        + (neur_j - c[1])**2)
27                   hij = alpha_t * exp(-1 * (dist_rij_rc)**2
                                / (2 * (eta_t**2)))
28                   input_to_w_old = (dataset_input[row][attr]
                                - trained_weight[neur_i][neur_j][attr])
29                   new_weight = (trained_weight[neur_i][neur_j][attr]
                                 + (hij * input_to_w_old))
30                   trained_weight[neur_i][neur_j][attr] = new_weight
31       return trained_weight
32
33
34    def training(dataset_input, input_weight, max_epoch, alpha_0,
                   eta_0):
35       trained_weight = input_weight
36
37       for t in range(1, max_epoch + 1):
38          alpha_t = alpha_0 * (1 / t)
39          eta_t = eta_0 * exp(-1 * (t / max_epoch))
40          trained_weight = one_epoch_training(
                          dataset_input,
                          trained_weight, alpha_t, eta_t)
41          print("Progress: {0}% AVG Silhouette Score:{1}".format(
                     (float(t) / max_epoch) * 100,
                     average_silhouette(trained_weight, dataset_input)
                   ), end='\r', flush=True)
42
43       return trained_weight
44
```

Kode Sumber 5.6 Proses *training* algoritme SOM

Terdapat 3 fungsi pada tahap *training*. Penjelasan dari fungsi-fungsi yang ada pada Kode Sumber 5.6 adalah sebagai berikut:

- Fungsi get_dist_data_from_neur

  Baris 4-5 : Perulangan sebanyak jumlah neuron yang ada pada bobot jaringan SOM.

  Baris 7 : Perulangan sebanyak jumlah atribut yang ada pada jaringan SOM.

  Baris 8 : Menghitung *squared distance* per atribut antara data *input* dengan bobot jaringan.

  Baris 9-10 : Setelah menjumlahkan masing-masing *squared distance* per atribut. Nilai jarak Euclidian didapatkan dari *root square* penjumlahan *squared distance*.

  Baris 10-11 : Nilai jarak Euclidian disimpan pada list dengan format *tuple* ((indeks baris neuron, indeks kolom neuron), jarak *Euclidian*) .

- Fungsi one_epoch_training

  Baris 15 : Melakukan iterasi sebanyak jumlah baris pada dataset *input.*

  Baris 16-21 : Baris kode ini merupakan baris untuk menentukan *best-matching-unit* pada algoritme SOM.

  Baris 18 : Memanggil fungsi get_dist_data_from_neur yang merupakan fungsi untuk mengambil data jarak antara data *input* dengan masing-masing neuron pada jaringan SOM beserta indeks neuronnya.

  Baris 19-20 : Mengambil indeks neuron yang memiliki jarak terpendek dengan data *input.*

  Baris 22-30 : Baris kode ini merupakan baris untuk melakukan proses *update* bobot pada jaringan SOM.

  Baris 23-25 : Melakukan perulangan sebanyak jumlah neuron dan jumlah atribut yang ada pada bobot jaringan SOM.

  Baris 26 : Menghitung jarak antara neuron pemenang dan neuron yang sedang terpilih.

  Baris 27 : Melakukan perhitungan fungsi h_{ij} yang melibatkan nilai *alpha* dan *eta.*

  Baris 28-29 : Melakukan perhitungan bobot baru yang melibatkan data pada atribut tertentu. Nilai bobot disimpan menggantikan bobot yang lama sebagai *output return value* fungsi.

Baris 31 : Nilai yang dikembalikan berupa bobot yang sudah dilatih sebanyak 1 *epoch.*

- Fungsi *training*

Baris 35 : Bobot yang akan dikembalikan memiliki nilai awal sama dengan bobot *input.*

Baris 37 : Melakukan perulangan sebanyak jumlah *epoch.*

Baris 38-39 : Menghitung nilai *alpha* dan *eta* sesuai dengan nilai t *epoch.*

Baris 40 : Menjalankan fungsi one_epoch_training dengan parameter *alpha* dan *eta* yang sudah dihitung berdasarkan nilai t *epoch*. Nilai yang dikembalikan adalah bobot jaringan yang sudah dilatih sebanyak 1 *epoch.*

Baris 41 : Mencetak *progress training* beserta nilainya.

Baris 43 : *Output return value* berupa bobot jaringan yang sudah dilatih sebanyak *epoch* yang telah ditentukan.

5.1.2.4 Implementasi fungsi proses penentuan *cluster* atau *clustering*

Fungsi penentuan_cluster pada dasarnya hanya memanggil fungsi get_dist_data_from_neur yang merupakan fungsi untuk mendapatkan data jarak. Kemudian, dari data jarak yang telah didapatkan tersebut diambil 1 neuron id yang memiliki jarak terpendek. Parameter *input* dari fungsi ini adalah bobot yang sudah di-*training* dan input data yang akan ditentukan id_neuron atau *cluster*-nya. Sedangkan *output return value* dari fungsi ini yaitu *tuple* baris dan kolom yang menjadi representasi posisi neuron yang terpilih. Implementasi dari fungsi penentuan *cluster* dapat dilihat pada Kode Sumber 5.7.

Algoritme 7: Fungsi menentukan cluster

```
1  def penentuan_cluster(trained_weight, input_data_row):
2        dist_data_to_neur = get_dist_data_from_neur(trained_weight,
                                    input_data_row)
3
4        # find best-matching-unit neuron
5        dist_sorted = sorted(dist_data_to_neur, key=lambda x: x[1])
6        c = dist_sorted[0][0]  # access tuple of neuron index
7
8        return c
9
```

Kode Sumber 5.7 Proses penentuan *cluster* atau *clustering*

Penjelasan dari Kode Sumber 5.5 adalah sebagai berikut:

Baris 2 : Menjalankan fungsi get_dist_data_from_neur yang merupakan fungsi untuk mendapatkan data jarak antara baris data input dengan neuron-neuron pada jaringan SOM.

Baris 5-6 : Melakukan pengurutan dan mengambil nilai indeks neuron yang memiliki jarak terpendek.

Baris 8 : Mengembalikan indeks neuron pemenang. Indeks neuron ini merupakan *tuple* dengan struktur (baris, kolom).

5.1.3 Implementasi kode program perhitungan nilai *Silhouette Coefficient*

Fungsi silhouette membutuhkan parameter *input* data bobot yang telah dilatih dan satu *dataset* utuh. *Output* dari fungsi ini adalah nilai bilangan desimal. Fungsi ini juga memanggil fungsi penentuan_cluster yang nantinya akan digunakan untuk menentukan id_cluster pada masing-masing baris data. Implementasi fungsi perhitungan *Silhouette Coefficient* dapat dilihat pada Kode Sumber 5.8. Pada implementasi tersebut, fungsi distance ditambahkan untuk membantu menentukan jarak antara dua titik data pada sebuah *dataset*.

Algoritme 8: Fungsi perhitungan Silhouette Coefficient
self_organizing_maps.py

```
1   def distance(dataset_input, data_point_a, data_point_b):
2       dist = 0
3       for attr in range(0, len(dataset_input[data_point_a])):
4           dist += (dataset_input[data_point_a][attr]
                        - dataset_input[data_point_b][attr]) ** 2
5       return sqrt(dist)
6
7
8   def silhouette(trained_weight, dataset_input):
9       cluster_result = dict()
10      cls_list = list()
11      for idx, x_data in enumerate(dataset_input):
12          c = penentuan_cluster(trained_weight, x_data)
13          cls_id = str(c[0]) + ';' + str(c[1]) + ';'
14          if cls_id not in cls_list:
15              cls_list.append(cls_id)
16              cluster_result[cls_id] = list()
17          cluster_result[cls_id].append(idx)
18
19      silhouette_score_data = dict()
20      avg = 0
21      for ci_id, ci in cluster_result.items():
22          silhouette_cluster = dict()
23          for i in ci:
24              a = 0
25              for j in ci:
26                  if i != j:
27                      a += distance(dataset_input, i, j)
28              a = 0 if (len(ci) <= 1) else (a / (len(ci) - 1))
29
30              cj_other_cluster = list()
31              for cj_id, cj in cluster_result.items():
32                  if ci_id != cj_id:
33                      b_temp = 0
34                      for j in cj:
35                          b_temp += distance(dataset_input, i, j)
36                      b_temp = b_temp / len(cj)
37                      cj_other_cluster.append(b_temp)
38              b = min(cj_other_cluster)
```

```
39              sil = 0 if (len(ci) <= 1) else (b - a) / max([a, b])
40              avg += sil
41              silhouette_cluster[i] = sil
42              # print("Silhouette data", i,
                            "Cluster ", ci_id, ":", sil)
43          silhouette_score_data[ci_id] = sorted(
                silhouette_cluster.items(), key=lambda x: x[1])
44  # print("-----------------------------------------------------")
45      avg /= len(dataset_input)
46      silhouette_score_data['avg'] = avg
47      return silhouette_score_data
48
```

Kode Sumber 5.8 Menghitung nilai *Silhouette Coefficient*

Penjelasan dari Kode Sumber 5.5 adalah sebagai berikut:

Baris 1-7 : Fungsi pembantu distance yang berfungsi untuk menentukan jarak *Euclidian* antara dua titik data.

Baris 8-10 : Fungsi silhouette membutuhkan parameter bobot jaringan SOM dan satu dataset utuh.

Baris 11-17 : Melakukan *clustering* per dataset dengan memanggil fungsi penentuan_cluster kemudian mengelompokkannya *dataset* berdasarkan *cluster* yang terbentuk.

Baris 22-29 : Melakukan perulangan untuk setiap *dataset* yang ada. Hal pertama yang dilakukan yaitu menghitung nilai a(i) yang merupakan jarak data dengan data lain pada satu *cluster*.

Baris 31-37 : Menghitung kandidat nilai b(i) yang merupakan *list* atau kumpulan dari jarak *dataset* ke masing-masing *cluster* yang terbentuk.

Baris 38 : Nilai b(i) merupakan jarak terpendek ke sebuah *cluster.*

Baris 39 : Menghitung nilai *Silhouette Coefficient* untuk data yang terpilih. Jika jumlah anggota *cluster* adalah 1 atau hanya dirinya sendiri, maka *Silhouette Coefficient* dari data tersebut bernilai 0.

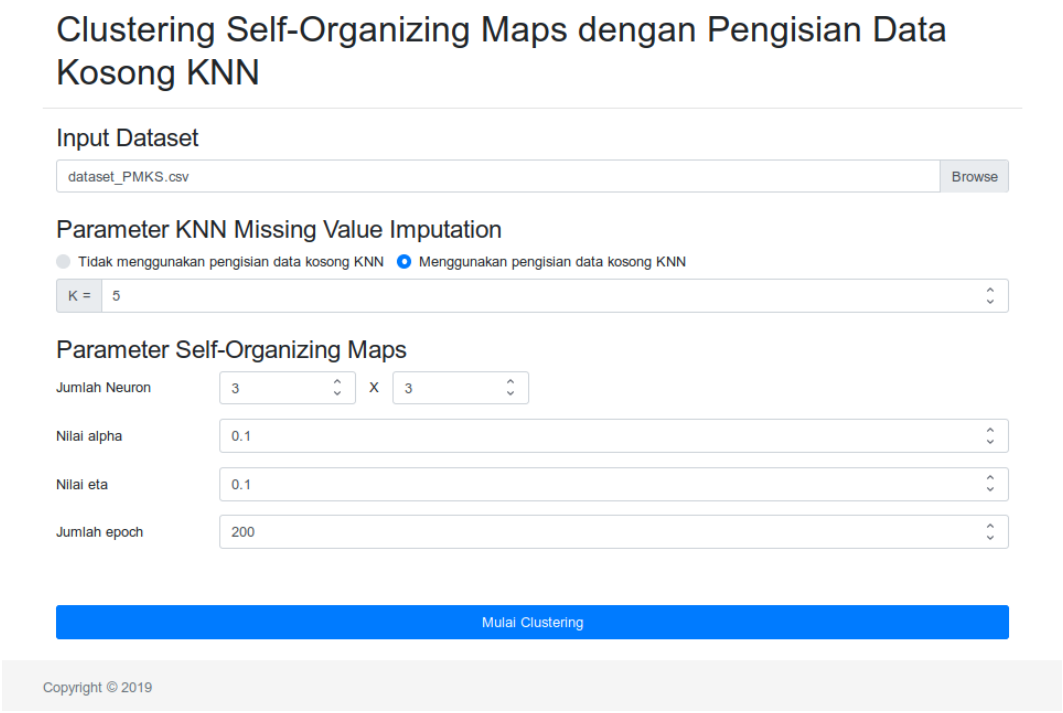
Baris 40-47 : Masing-masing nilai *Silhouette Coefficient* untuk setiap data akan disimpan di sebuah *list*. Kemudian di dalam *list* tersebut juga terdapat nilai rata-rata *Silhouette Coefficient* secara global untuk semua objek data. Sehingga nilai kembalian dari fungsi adalah nilai *Silhouette Coefficient* masing-masing *dataset* dan rata-rata *Silhouette Coefficient*.

5.2 Implementasi Antarmuka

Tahap ini merupakan implementasi dari rancangan antarmuka yang telah didefinisikan di bab 4. Rancangan tersebut diimplementasikan dalam bentuk halaman *web*. Terdapat 2 halaman *web* yang diimplementasikan, yaitu antarmuka mulai *clustering* serta antarmuka hasil *clustering*.

5.2.1 Implementasi antarmuka mulai *clustering*

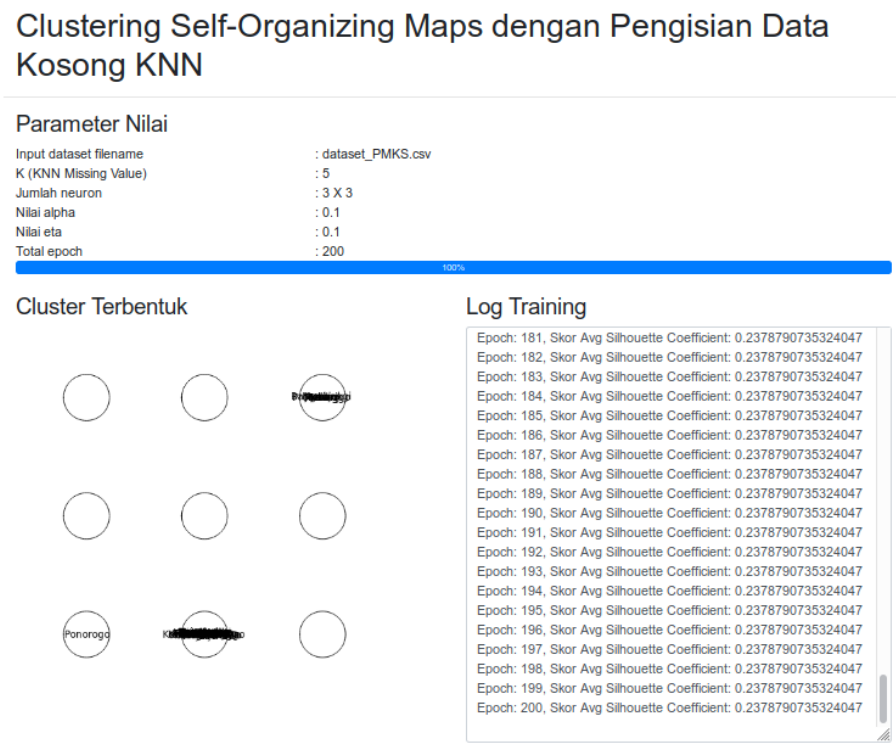
Pada halaman mulai *clustering*, pengguna disediakan *form* yang berisi *input file dataset* dan beberapa *input* nilai parameter algoritme. Pengguna diharapkan untuk memilih *dataset* dengan format .csv. Kemudian setelah memilih *dataset*, pengguna akan mengisi nilai parameter algoritme. Setelah itu pengguna menekan tombol mulai *clustering* untuk memulai proses *clustering*. Tampilan antarmuka mulai *clustering* dapat dilihat pada Gambar 5.1.



Gambar 5.1 Implementasi antarmuka mulai *clustering*

5.2.2 Implementasi antarmuka hasil *clustering*

Setelah pengguna menekan tombol mulai *clustering*, maka pengguna akan disediakan tampilan *progress* dari *clustering*. Pengguna diberikan tampilan berupa nilai parameter yang telah dimasukkan, bentuk *cluster* yang terbentuk, log *training* serta detail dari *cluster* yang terbentuk. Tampilan hasil *clustering* dapat dilihat pada Gambar 5.2.



Gambar 5.2 Implementasi antarmuka hasil *clustering*

Gambar 5.2 Implementasi antarmuka hasil *clustering* (lanjutan)

BAB 6 PENGUJIAN DAN ANALISIS HASIL

6.1 Pengujian Parameter Algoritme

Hasil pengujian algoritme akan ditampilkan dalam bentuk grafik. Proses pengujian parameter nilai pada masing-masing algoritme dilakukan sebanyak 10 kali pada masing-masing nilai uji dan dicari nilai rata-rata pada nilai uji tersebut. Grafik akan menampilkan nilai rata-rata dan nilai *Silhouette Coefficient* tertinggi dari 10 kali percobaan yang dilakukan. Representasi nilai pada pengujian akan tergantung pada jenis pengukuran sesuai dengan apa yang telah ditentukan pada proses perancangan skenario pengujian.

6.1.1 Pengujian parameter pada Algoritme *Self-Organizing Maps* (SOM)

Pengujian ini dilakukan untuk mengetahui parameter algoritme SOM terbaik yang akan digunakan untuk kasus *clustering* pada penelitian kali ini. Sebuah algoritme SOM terdiri dari beberapa parameter yang harus diujikan yaitu (1) konstanta *learning rate* / *alpha*, (2) konstanta signifikansi *update* bobot atau eta, (3) jumlah *epoch* dan (4) jumlah neuron. Masing-masing pengujian diukur menggunakan *average Silhouette Coefficient.*

6.1.1.1 Hasil pengujian parameter konstanta *alpha* pada algoritme SOM

Pengujian ini dilakukan untuk mengetahui nilai *learning rate* atau *alpha* yang memberikan nilai *Silhouette Coefficient* tertinggi. Nilai *alpha* yang diujikan yaitu nilai *alpha* dengan rentang antara 0,1 hingga 1 dengan kelipatan kenaikan 0,1.

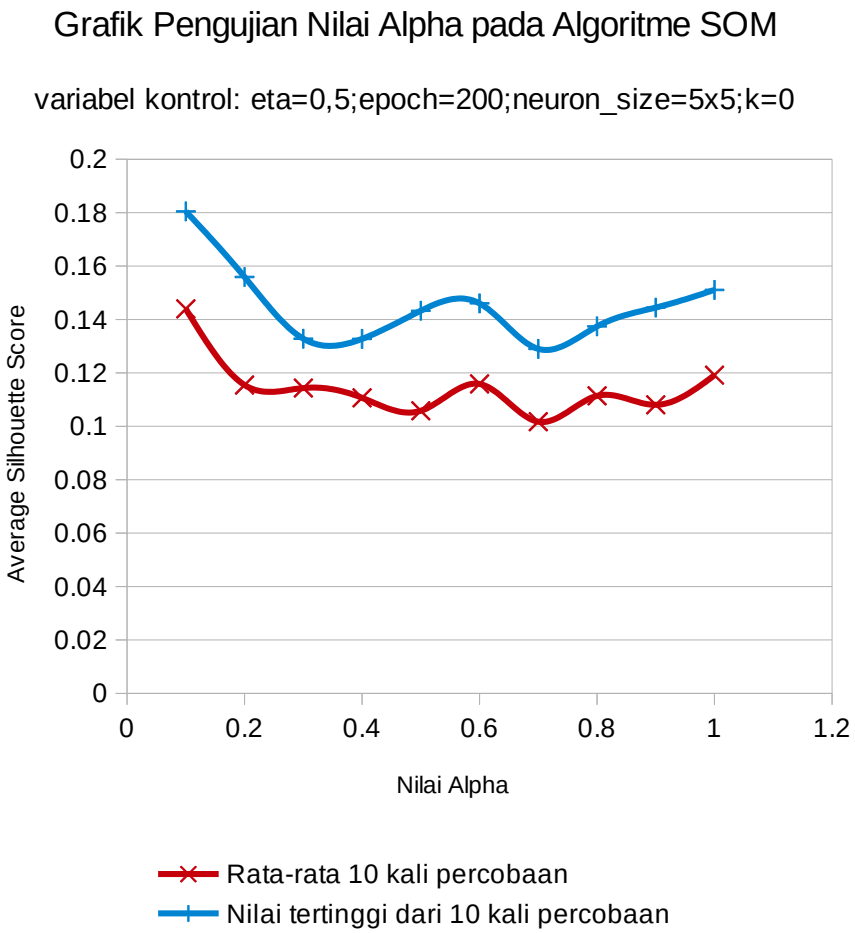
Tabel 6.1 Hasil pengujian konstanta nilai *alpha* pada algoritme SOM

Nilai *alpha*	*Silhouette Coefficient*										Rata-rata	Nilai tertinggi
	1	2	3	4	5	6	7	8	9	10		
0,1	0,138	0,160	0,179	0,127	0,122	0,096	0,145	0,145	0,180	0,149	0,144	0,180
0,2	0,120	0,114	0,156	0,122	0,141	0,091	0,054	0,112	0,131	0,112	0,115	0,156
0,3	0,107	0,076	0,133	0,100	0,126	0,092	0,122	0,133	0,126	0,128	0,114	0,133
0,4	0,086	0,104	0,118	0,091	0,125	0,103	0,107	0,130	0,133	0,109	0,111	0,133
0,5	0,098	0,101	0,087	0,039	0,119	0,113	0,143	0,122	0,131	0,104	0,106	0,143
0,6	0,129	0,146	0,094	0,135	0,076	0,101	0,125	0,115	0,100	0,137	0,116	0,146
0,7	0,037	0,129	0,118	0,116	0,116	0,095	0,091	0,085	0,121	0,110	0,102	0,129
0,8	0,131	0,110	0,100	0,117	0,137	0,070	0,121	0,102	0,103	0,123	0,111	0,137

Tabel 6.1 Hasil pengujian konstanta nilai *alpha* pada algoritme SOM (lanjutan)

Nilai *alpha*	*Silhouette Coefficient*										Rata-rata	Nilai tertinggi
	1	2	3	4	5	6	7	8	9	10		
0,9	0,117	0,082	0,079	0,100	0,113	0,128	0,144	0,122	0,129	0,065	0,108	0,144
1	0,139	0,123	0,123	0,077	0,129	0,115	0,098	0,151	0,129	0,107	0,119	0,151

Hasil dari pengujian nilai *alpha* beserta rata-rata dan nilai tertinggi dari 10 kali pengujian dapat dilihat pada Tabel 6.1. Proses *clustering* cenderung memberikan hasil *average Silhouette Coefficient* tertinggi ketika konstanta *alpha* atau *learning rate* bernilai 0.1. Nilai *alpha* 0.1 memberikan hasil *average Silhouette Coefficient* sebesar 0.144 untuk rata-rata 10 kali percobaan. Selain itu, proses *clustering* juga mencapai nilai tertinggi *average Silhouette Coefficient* ketika konstanta *alpha* bernilai 0,1 dengan nilai evaluasi 0,180.



Gambar 6.1 Grafik pengujian nilai *alpha* pada algoritme SOM

Pada grafik pengujian yang ditunjukkan oleh Gambar 6.1, semakin bertambahnya nilai *alpha* yang digunakan pada algoritme SOM, nilai *Silhouette*

Coefficient sebagai evaluasi hasil *cluster* memiliki *trend-line* yang cenderung menurun baik pada rata-rata 10 kali percobaan (garis merah) maupun nilai tertinggi dari 10 kali percobaan (garis biru). Nilai *alpha* yang cenderung besar, cenderung memberikan *update* bobot yang juga begitu besar yang mana dapat mempengaruhi keanggotaan *cluster* yang terbentuk. Sehingga dalam pengujian ini, nilai *alpha* yang terlalu tinggi dapat mempengaruhi keanggotaan *cluster* yang mana juga mempengaruhi skor *Silhouette Coefficient*.

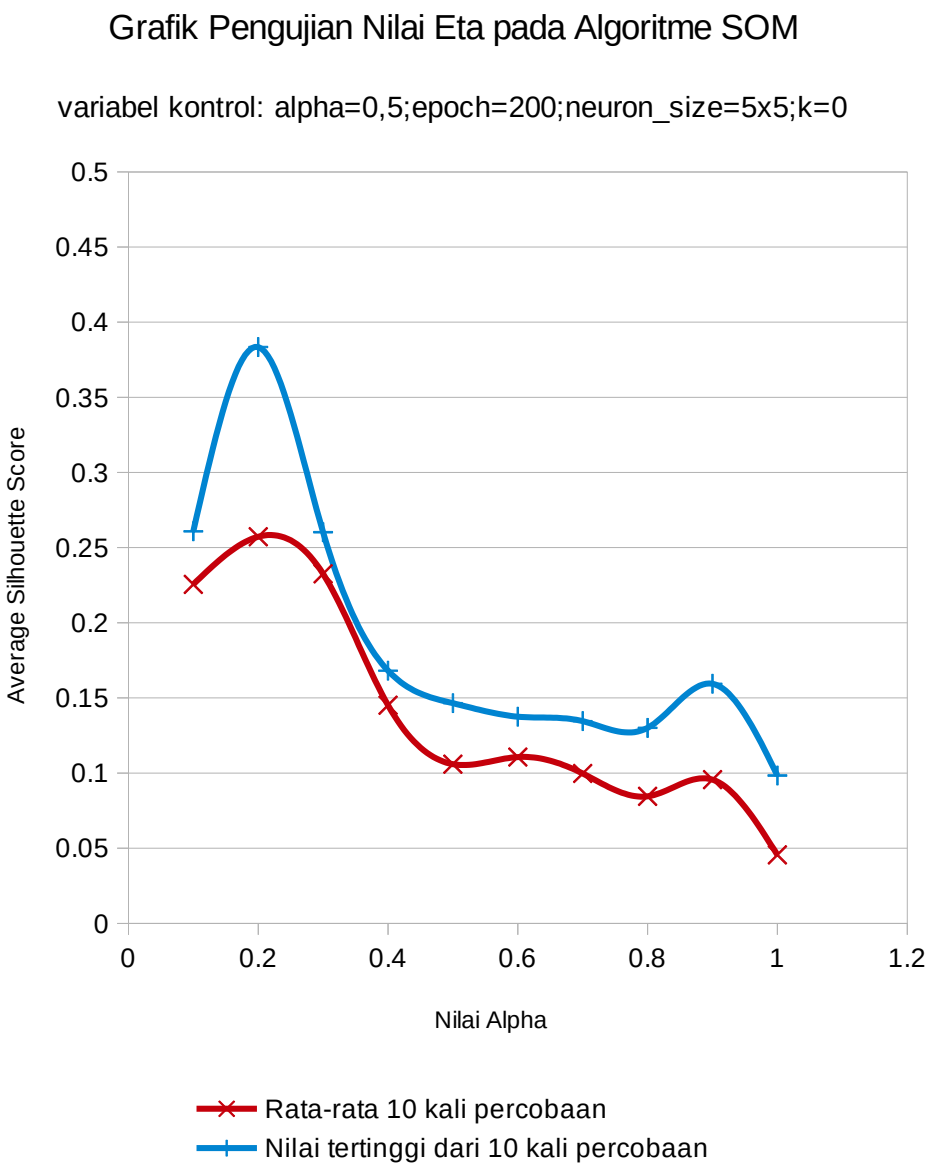
6.1.1.2 Hasil pengujian parameter konstanta *eta* pada algoritme SOM

Pengujian ini dilakukan untuk mengetahui nilai eta atau radius *update* ketetanggaan neuron selama proses *training* jaringan SOM berlangsung. Sama seperti pengujian nilai *alpha*, nilai eta yang akan diujikan yaitu nilai *alpha* dengan rentang antara 0.1 hingga 1 dengan kelipatan kenaikan 0.1.

Tabel 6.2 Hasil pengujian konstanta nilai *eta* pada algoritme SOM

Nilai *eta*	*Silhouette Coefficient*										Rata-rata	Nilai tertinggi
	1	2	3	4	5	6	7	8	9	10		
0,1	0,205	0,196	0,261	0,257	0,184	0,184	0,195	0,257	0,257	0,259	0,226	0,261
0,2	0,247	0,256	0,202	0,383	0,257	0,279	0,187	0,250	0,255	0,257	0,257	0,383
0,3	0,257	0,260	0,179	0,244	0,247	0,182	0,196	0,259	0,257	0,244	0,233	0,260
0,4	0,150	0,127	0,144	0,160	0,168	0,137	0,147	0,110	0,158	0,149	0,145	0,168
0,5	0,126	0,121	0,083	0,110	0,147	0,084	0,100	0,054	0,108	0,126	0,106	0,147
0,6	0,116	0,082	0,091	0,138	0,065	0,134	0,127	0,117	0,120	0,116	0,111	0,138
0,7	0,064	0,120	0,085	0,107	0,135	0,096	0,092	0,102	0,124	0,074	0,100	0,135
0,8	0,050	0,074	0,086	0,045	0,053	0,118	0,130	0,121	0,082	0,086	0,085	0,130
0,9	0,048	0,052	0,062	0,133	0,107	0,128	0,087	0,073	0,106	0,159	0,096	0,159
1	0,027	0,052	0,081	0,038	-0,007	0,057	0,098	0,054	-0,032	0,087	0,046	0,098

Hasil dari pengujian nilai eta beserta rata-rata dan nilai tertinggi dari 10 kali pengujian dapat dilihat pada Tabel 6.2. Jaringan SOM cenderung memberikan hasil *Silhouette Coefficient* yang cukup tinggi ketika jaringan SOM di-*training* menggunakan konstanta *eta* dengan nilai 0,2. Nilai 0,2 pada konstanta eta memberikan hasil *Silhouette Coefficient* sebesar 0,257 untuk rata-rata 10 kali percobaan. Selain itu, nilai 0,2 juga memberikan hasil *Silhouette Coefficient* paling tinggi yang dapat dicapai yaitu sebesar 0,383 dibandingkan dengan nilai *eta* yang lain.



Gambar 6.2 Grafik pengujian nilai *eta* pada algoritme SOM

Dari grafik pengujian yang terdapat pada Gambar 6.2, dapat dilihat bahwa konstanta *eta* yang memberikan hasil tertinggi *Silhouette Coefficient* yaitu ketika *eta* bernilai 0,2 baik untuk rata-rata 10 percobaan (garis warna merah) maupun nilai tertingginya (garis warna biru). Terlalu tinggi nilai *eta* menyebabkan neuron-neuron tetangga dari neuron pemenang yang seharusnya tidak dilakukan *update* bobot (neuron yang tidak relevan) menjadi ter-*update*. Terlalu kecil nilai *eta* yang digunakan juga menyebabkan neuron tetangga yang seharusnya bersifat korporatif dengan neuron pemenang menjadi tidak ter-*update*. Nilai *eta* yang ideal adalah 0,2 pada pengujian ini.

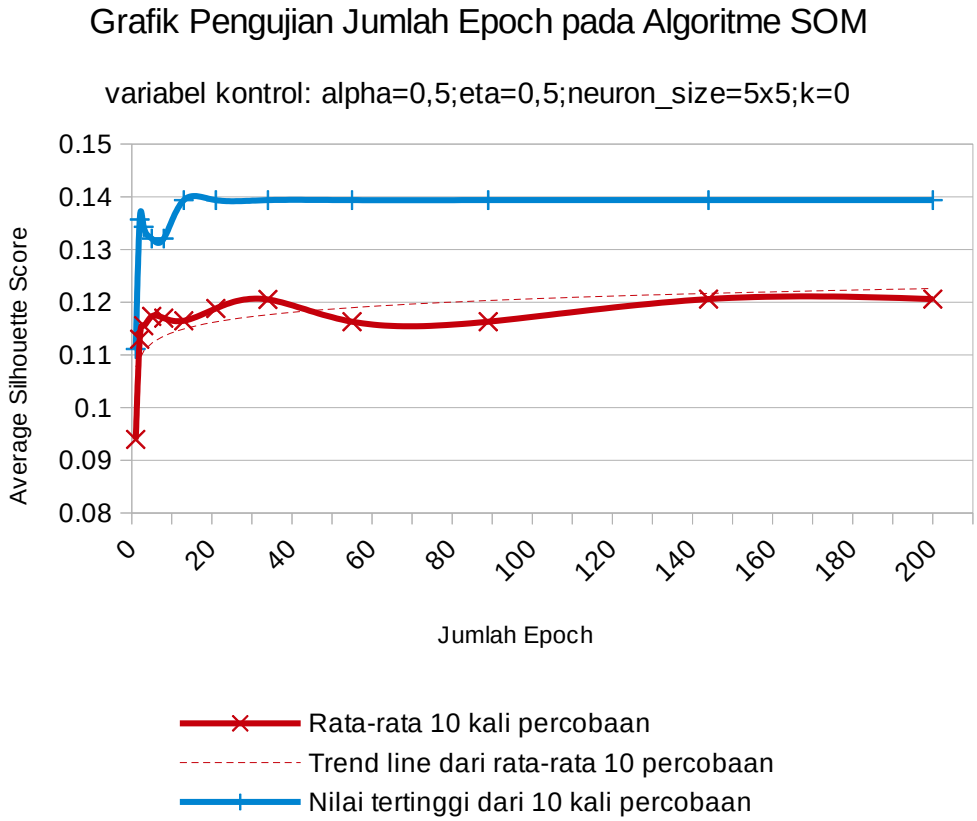
6.1.1.3 Hasil pengujian parameter jumlah *epoch* pada algoritme SOM

Pengujian jumlah *epoch* digunakan untuk mengetahui parameter *epoch* atau jumlah iterasi yang memberikan hasil signifikan pada proses *clustering* ketika diukur menggunakan *Silhouette Coefficient*. Pengujian dilakukan sebanyak 10 kali untuk tiap parameter nilai jumlah *epoch*. Nilai yang diujikan yaitu antara 1 hingga 200 dengan kelipatan kenaikan mengikuti deret *Fibonacci*.

Tabel 6.3 Hasil pengujian parameter jumlah epoch pada algoritme SOM

Jumlah *epoch*	*Silhouette Coefficient*										Rata-rata	Nilai tertinggi
	1	2	3	4	5	6	7	8	9	10		
1	0,077	0,064	0,101	0,106	0,071	0,105	0,093	0,076	0,097	0,100	0,089	0,106
2	0,110	0,062	0,113	0,131	0,083	0,107	0,127	0,128	0,114	0,105	0,108	0,131
3	0,129	0,060	0,115	0,127	0,082	0,119	0,129	0,123	0,115	0,105	0,111	0,129
5	0,118	0,082	0,098	0,123	0,104	0,112	0,127	0,123	0,126	0,109	0,112	0,127
8	0,118	0,100	0,096	0,123	0,109	0,112	0,127	0,109	0,126	0,099	0,112	0,127
13	0,118	0,100	0,098	0,134	0,109	0,110	0,127	0,091	0,128	0,099	0,111	0,134
21	0,118	0,100	0,098	0,134	0,109	0,117	0,127	0,091	0,128	0,116	0,114	0,134
34	0,118	0,100	0,116	0,134	0,109	0,117	0,127	0,091	0,128	0,116	0,116	0,134
55	0,118	0,100	0,073	0,134	0,109	0,117	0,127	0,091	0,128	0,116	0,111	0,134
89	0,118	0,100	0,073	0,134	0,109	0,117	0,127	0,091	0,128	0,116	0,111	0,134
144	0,118	0,100	0,116	0,134	0,109	0,117	0,127	0,091	0,128	0,116	0,116	0,134
200	0,118	0,100	0,116	0,134	0,109	0,117	0,127	0,091	0,128	0,116	0,116	0,134

Hasil pengujian parameter jumlah *epoch* dapat dilihat pada Tabel 6.3. Dari tabel tersebut, jumlah *epoch* 34, 144 dan 200 memberikan nilai rata-rata tertinggi *Silhouette Coefficient* sebesar 0.166 untuk 10 kali percobaan. Sedangkan untuk beberapa kasus percobaan, terdapat juga percobaan yang memberikan nilai tertinggi dan konvergen untuk jumlah *epoch* yang lebih sedikit. Misalnya, percobaan nomor 4 memberikan hasil rata-rata 10 percobaan *Silhouette Coefficient* tertinggi dan konvergen sebesar 0,134 ketika jumlah *epoch* bernilai 34. Dalam hal ini, iterasi dilanjutkan hingga mencapai titik dimana proses *training* memiliki nilai *Silhouette Coefficient* yang stabil atau konvergen.



Gambar 6.3 Grafik pengujian jumlah *epoch* pada algoritme SOM

Pada grafik yang ditunjukkan pada Gambar 6.3, pengujian jumlah *epoch* yang dilakukan sudah memberikan hasil yang konvergen pada *epoch* ke-34. Namun untuk beberapa kasus percobaan, terdapat percobaan yang memberikan penurunan nilai *Silhouette Coefficient* pada *epoch* ke-55 dan 89. Konvergensi sempurna untuk 10 percobaan baru bisa dicapai ketika jumlah epoch bernilai 160. Setelah nilai 160, tidak ada lagi perubahan pada nilai rata-rata *Silhouette Coefficient*. Fungsi linear penurunan nilai *alpha* yang bergantung *epoch* (*alpha* * 1/t) yang digunakan membuat proses *update* bobot untuk setiap *epoch*-nya semakin mengecil karena *alpha* yang digunakan juga semakin kecil. Nilai *alpha* yang kecil ini membuat proses *update* bobot pada jaringan SOM tidak terlalu berlebihan ketika jumlah *epoch* semakin besar.

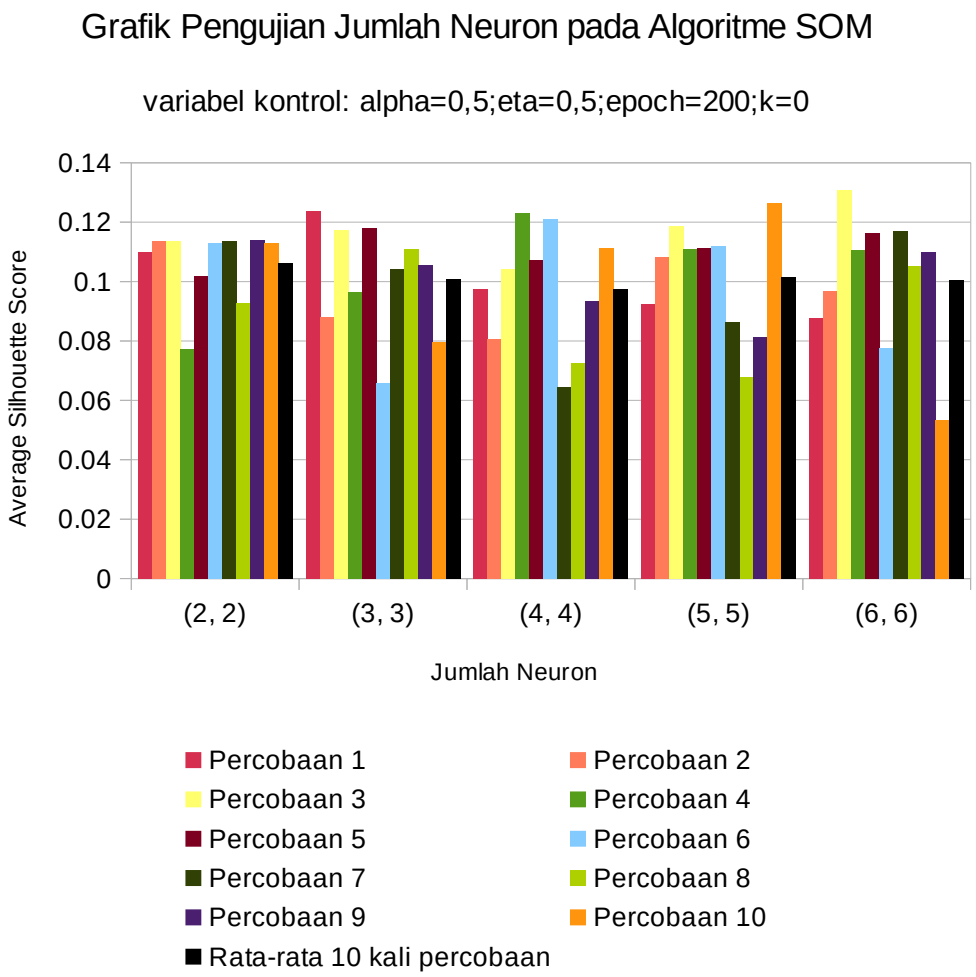
6.1.1.4 Hasil pengujian parameter jumlah *neuron* pada algoritme SOM

Pengujian parameter jumlah neuron dilakukan untuk mengetahui jumlah neuron yang sesuai untuk proses *clustering* data PMKS. Karena arsitektur jaringan SOM yang digunakan adalah 2D-*Rectangular*, maka nilai parameter jumlah neuron yang digunakan mengikuti deret geometri yaitu (2x2), (3x3), (4x4), (5x5) dan (6x6). Evaluasi proses pengujian menggunakan *average Silhouette Coefficient.* Masing-masing parameter nilai diuji sebanyak 10 kali dan diambil rata-rata *Silhouette Coefficient*-nya.

Tabel 6.4 Hasil pengujian jumlah neuron pada algoritme SOM

Jumlah neuron	*Silhouette Coefficient*										Rata-rata
	1	2	3	4	5	6	7	8	9	10	
(2, 2)	0,110	0,114	0,114	0,077	0,102	0,113	0,114	0,093	0,114	0,113	0,106
(3, 3)	0,124	0,088	0,117	0,096	0,118	0,066	0,104	0,111	0,105	0,080	0,101
(4, 4)	0,097	0,081	0,104	0,123	0,107	0,121	0,064	0,072	0,093	0,111	0,098
(5, 5)	0,092	0,108	0,119	0,111	0,111	0,112	0,086	0,068	0,081	0,127	0,102
(6, 6)	0,088	0,097	0,131	0,110	0,116	0,078	0,117	0,105	0,110	0,053	0,101

Hasil pengujian jumlah neuron secara detail dapat dilihat pada Tabel 6.4. Pada pengujian tersebut, parameter jumlah neuron sebesar 2x2 memberikan nilai-rata *Silhouette Coefficient* tertinggi dengan nilai 0,106.



Gambar 6.4 Grafik pengujian jumlah neuron pada algoritme SOM

Gambar 6.4 menunjukkan grafik hasil pengujian jumlah neuron untuk 10 kali percobaan pada masing-masing parameter yang diujikan. Bar yang berwarna hitam merupakan rata-rata dari 10 percobaan tersebut. Pada algoritme SOM, jumlah neuron merepresentasikan kandidat *cluster* yang akan terbentuk. Ketika jumlah neuron pada jaringan SOM berjumlah sangat banyak, objek data memiliki kecenderungan untuk menyebar ke seluruh neuron-neuron yang ada. Penyebaran ini memungkinkan terbentuknya *singleton cluster* atau *cluster* yang memiliki 1 anggota. Karena pada evaluasi *Silhouette Coefficaient* sebuah *singleton* memiliki nilai *Silhouette Coefficient* sebesar 0, maka hal ini mempengaruhi nilai *global average Silhouette Coefficient* yang cenderung turun.

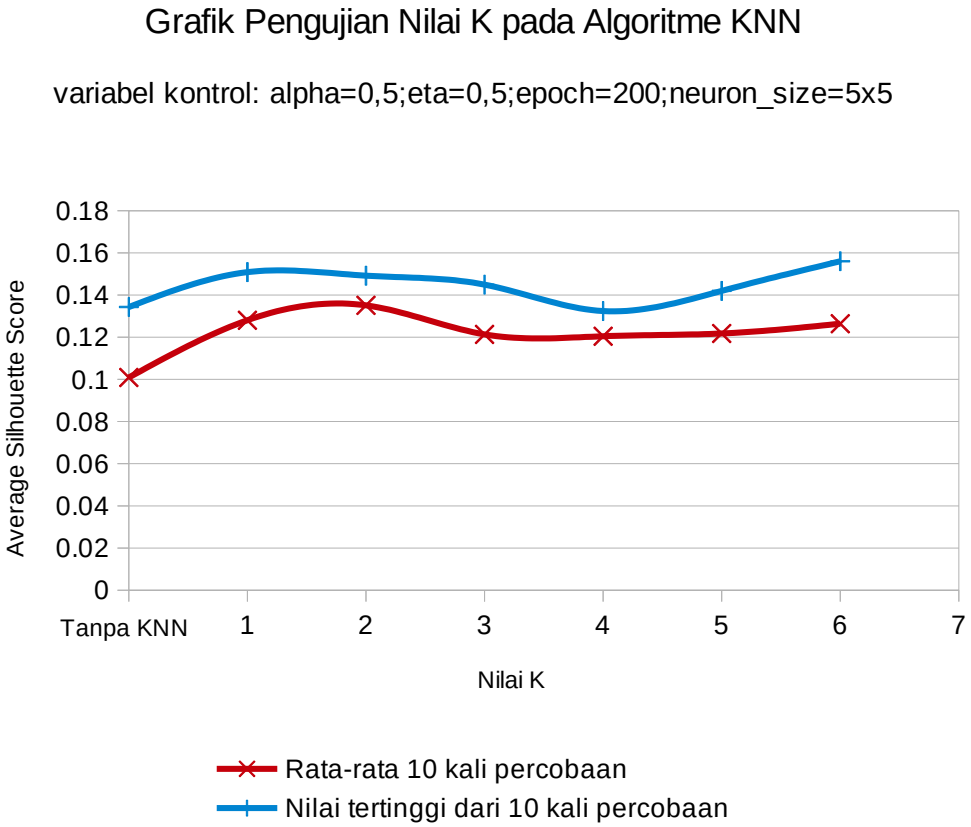
6.1.2 Hasil pengujian nilai *K* pada algoritme *K-Nearest Neighbors* (KNN)

Pengujian nilai *K* pada algoritme KNN digunakan untuk mengetahui pengaruh imputasi atau pengisian data kosong pada kualitas *cluster* yang terbentuk. Nilai *K* yang diujikan yaitu tidak menggunakan pengisian data kosong KNN kemudian menggunakan pengisian data kosong KNN dengan *K* 1, 2, 3, 4, 5 dan 6.

Tabel 6.5 Hasil pengujian parameter nilai *K* pada algoritme KNN

Nilai K	*Silhouette Coefficient*										Rata-rata	Nilai tertinggi
	1	2	3	4	5	6	7	8	9	10		
Tanpa KNN	0,060	0,066	0,106	0,109	0,115	0,103	0,084	0,109	0,124	0,134	0,101	0,134
1	0,151	0,080	0,127	0,131	0,146	0,111	0,108	0,141	0,141	0,145	0,128	0,151
2	0,149	0,130	0,136	0,136	0,129	0,136	0,117	0,138	0,146	0,134	0,135	0,149
3	0,135	0,133	0,145	0,115	0,129	0,081	0,119	0,095	0,136	0,125	0,121	0,145
4	0,129	0,125	0,126	0,123	0,119	0,128	0,088	0,108	0,125	0,132	0,120	0,132
5	0,108	0,117	0,115	0,131	0,142	0,112	0,127	0,102	0,122	0,142	0,122	0,142
6	0,135	0,121	0,093	0,124	0,156	0,123	0,136	0,117	0,122	0,136	0,126	0,156

Hasil pengujian nilai *K* tersebut dapat dilihat pada Tabel 6.5. Dalam hal ini, *K* dengan nilai 2 memiliki rata-rata *Silhouette Coefficient* tertinggi dibandingkan dengan *K* yang bernilai 0 (tidak menggunakan pengisian data kosong) maupun ketika dibandingkan dengan nilai *K* yang lain. *K* yang bernilai 2 memberikan nilai rata-rata *Silhouette Coefficient* tertinggi dengan nilai sebesar 0,135. Sedangkan pada *clustering* tanpa pengisian data kosong KNN, rata-rata *Silhouette Coefficient* yang didapatkan yaitu 0,101.



Gambar 6.5 Grafik pengujian nilai K pada algoritme KNN

Dari grafik yang ditunjukkan pada Gambar 6.5, penggunaan algoritme KNN untuk pengisian data kosong memberikan hasil peningkatan skor *Silhouette Coefficient*. Pengaruh nilai *K* di sini yaitu terdapat pada data-data relevan yang akan digunakan untuk pengisian data kosong. Dalam hal ini, nilai *K* yang terlalu besar membuat data-data yang tidak relevan untuk mengisi sebuah data kosong menjadi diikutsertakan dalam perhitungan untuk pengisian data kosong. Nilai *K* yang terlalu kecil menyebabkan data-data relevan yang seharusnya menjadi kandidat untuk mengisi sebuah data kosong menjadi tidak diperhitungkan. Dari 6 parameter nilai *K* yang diujikan, Nilai *K* yang bernilai 2 memberikan hasil ideal untuk rata-rata dari 10 percobaan dengan *global average Silhouette Coefficient* tertinggi. *K* yang bernilai 2 juga memberikan peningkatan nilai *Silhouette Coefficient* dari 0,101 hingga 0,135 (sekitar 3,4%).

6.2 Analisis Hasil Pengujian

Sub-bab analisis hasil pengujian membahas mengenai bagaimana parameter-parameter terbaik yang telah didapatkan ketika digunakan untuk proses *clustering*. *Cluster* terbentuk dengan nilai *Silhouette Coefficient* tertinggi akan dipilih untuk dilakukan analisis *cluster*. Analisis *cluster* yang dilakukan yaitu membandingkan perbedaan dari variabel / atribut data dan mengambil lima atribut teratas dengan perbedaan tertinggi. Data yang digunakan untuk analisis

cluster yaitu data yang sudah dilakukan pengisian data kosong dan sudah dinormalisasi.

Dari hasil pengujian, diperoleh parameter terbaik dan optimal sebaik berikut:

- Nilai *learning rate* / *alpha*: 0,1
- Nilai signifikansi update bobot / eta: 0,2
- Jumlah Epoch: 160
- Jumlah Neuron: 2x2
- Nilai *K*: 2

Setelah dilakukan percobaan untuk parameter tersebut sebanyak lima kali. Tahap analisis akan mengambil bentuk *cluster* yang memiliki nilai *Silhouette Coefficient* tertinggi kemudian melakukan analisis pada *cluster* tersebut.

Tabel 6.6 Hasil *Silhouette Coefficient* dari 5 kali percobaan

Percobaan ke-	*Global average Silhouette Coefficient*	*Cluster dihasilkan*	Proporsi *Cluster*
1	0,250	3	1:2:35
2	**0,351**	**2**	**1:37**
3	0,231	3	1:1:36
4	0,295	2	3:35
5	0,258	2	5:33

Tabel 6.6 menunjukkan nilai *Silhouette Coefficient* yang didapatkan untuk lima kali percobaan dengan menggunakan parameter terbaik yang didapatkan pada pengujian sebelumnya. Setelah melakukan proses *clustering* dengan menggunakan parameter-parameter tersebut, didapatkan hasil tertinggi nilai *Silhouette Coefficient* sebesar 0,351. *Cluster* yang dihasilkan yaitu sebanyak 2.

Tabel 6.7 Hasil *clustering* untuk semua wilayah pada percobaan ke-2

Cluster (0, 1)	Pacitan	Ponorogo	Trenggalek	Tulungagung
	Blitar	Kediri	Malang	Lumajang
	Banyuwangi	Bondowoso	Situbondo	Probolinggo
	Pasuruan	Sidoarjo	Mojokerto	Jombang
	Nganjuk	Madiun	Magetan	Ngawi
	Bojonegoro	Tuban	Lamongan	Gresik

Tabel 6.7 Hasil *clustering* untuk semua wilayah pada percobaan ke-2 (lanjutan)

	Bangkalan	Sampang	Pameksan	Sumenep
	Kota Kediri	Kota Blitar	Kota Malang	Kota Probolinggo
	Kota Pasuruan	Kota Mojokerto	Kota Madiun	Kota Surabaya
	Kota Batu			
Cluster (0, 0)	Jember			

Pada Tabel 6.7, proses *clustering* memberikan hasil terdapat 1 *cluster singleton*. Sedangkan sisa objek data mengumpul ke satu tempat sehingga terbentuk *cluster-cluster* dengan proporsi 1:37 dari 38 data. *Cluster singleton* tersebut beranggotakan kabupaten Jember. *Cluster singleton* ini merupakan *outlier* data.

Tabel 6.8 Analisis per atribut data pada hasil *clustering* pada percobaan ke-2

Variabel / Atribut Data	**Cluster (0,0)**	**Cluster (0,1)**	**Selisih**
Lanjut Usia Terlantar	1,00	0,21	0,79
Gelandangan Dan Gelandangan Psikotik	0,99	0,21	0,78
Pemulung	1,00	0,25	0,75
Pengemis	1,00	0,26	0,74
Kelompok Minoritas	0,97	0,25	0,73
Tuna Susila	1,00	0,34	0,66
Anak Dengan Kedisabilitasan	0,79	0,23	0,56
Korban Penyalagunaan Napza	0,63	0,12	0,51
Orang Dengan HIV	0,67	0,17	0,50
Korban Tindak Kekerasan Atau Yang Diperlakukan Salah	0,81	0,31	0,50
Anak Yang Menjadi Korban Tindak Kekerasan Atau Yang Diperlakukan Salah	0,78	0,34	0,44
Anak Berhadapan Dengan Hukum	0,47	0,20	0,27

Tabel 6.8 Analisis per atribut data pada hasil *clustering* pada percobaan ke-2 (lanjutan)

Variabel / Atribut Data	Cluster (0,0)	Cluster (0,1)	Selisih
Anak Jalanan	0,39	0,14	0,25
Anak Yang Memerlukan Perlindungan Khusus	0,41	0,17	0,25
Perempuan Rawan Sosial Ekonomi	0,48	0,26	0,23
Orang Dengan AIDS	0,41	0,20	0,21
Bekas Warga Binaan Lembaga Pemasyarakatan	0,29	0,11	0,18
Keluarga Bermasalah Sosial Psikologis	0,26	0,13	0,14
Masyarakat Daerah Tertinggal Dan Terpencil	0,11	0,20	0,09
Korban Trafficking	0,15	0,07	0,08
Anak Balita Terlantar	0,03	0,08	0,05
Penyandang Disabilitas & Bekas Penderita Penyakit Kronis	0,19	0,14	0,04
Korban Bencana Alam	0,01	0,04	0,03
Pekerja Migran Bermasalah Sosial	0,17	0,15	0,02
Korban Bencana Sosial / Pengungsi	0,06	0,08	0,02
Anak Terlantar	0,07	0,07	0,01

Tabel 6.8 merupakan analisis data dengan membandingkan rata-rata setiap anggota pada masing-masing *cluster*. Proses analisis menggunakan *dataset* yang telah dinormalisasi dan dilakukan pengisian data kosong menggunakan KNN dengan menggunakan parameter terbaik.

Pada tabel tersebut, dapat dilihat bahwa *cluster* (0,0) yang terdiri dari kabupaten Jember saja merupakan sebuah *outlier*. Data Kabupaten Jember memiliki perbedaan data PMKS yang cukup jauh dibandingkan dengan kabupaten-kabupaten lainnya. Lima atribut teratas yang memiliki selisih yang cukup jauh antara *cluster* (0,0) dan (0,1) adalah Lanjut Usia Terlantar, Gelandangan dan Gelandangan Psikotik, Pemulung, Pengemis dan Kelompok Minoritas. Lanjut Usia Terlantar memiliki selisih sebesar 0,79 yang merupakan selisih paling tertinggi dibanding dengan atribut yang lain.

Hasil *clustering* dengan *Silhouette Coefficient* 0,351 termasuk ke dalam kategori *weak structure.* Nilai kecil pada *Silhouette Coefficient* dapat terjadi karena selisih nilai *inter-cluster* b(i) dan nilai *intra cluster* a(i) tidak terlalu tinggi.

Sebuah *clustering* yang bagus normalnya memiliki proporsi selisih *inter-cluster* dan *intra-cluster* yang besar ketika dibagi dengan nilai *inter-cluster*-nya sendiri.

BAB 7 PENUTUP

Bagian penutup memuat kesimpulan yang berisi poin yang didapatkan dari penelitian serta saran untuk penelitian selanjutnya.

7.1 Kesimpulan

Kesimpulan membahas poin yang didapatkan pada bab Hasil dan Pembahasan yang telah dilakukan sebelumnya serta mencerminkan jawaban dari rumusan masalah yang telah ditentukan di bab pendahuluan. Berdasarkan pengujian serta analisis yang telah dilakukan sebelumnya, poin-poin yang didapatkan untuk penelitian kali ini sebagai berikut:

1. Berdasarkan pengujian yang dilakukan untuk masing-masing parameter ketika diukur dengan *Silhouette Coefficient*, parameter terbaik untuk algoritme *Self-Organizing Maps* ketika digunakan untuk pengelompokan data PMKS yaitu (1) *alpha* bernilai 0,1; (2) *eta* bernilai 0,2; (3) jumlah epoch bernilai 160 dan (4) jumlah neuron dengan ukuran 2x2. Sedangkan pada algoritme *K-Nearest Neighbors*, parameter untuk nilai *K* terbaik yang didapatkan yaitu 2.
2. Imputasi data atau pengisian data kosong menggunakan *K-Nearest Neighbors* memberikan pengaruh peningkatan nilai *Silhouette Coefficient* sebesar 3,4%. dibandingkan dengan tanpa pengisian data kosong. Pada pengujian nilai K untuk *K-Nearest Neighbors*, *K* yang bernilai 2 memiliki nilai *Silhouette Coefficient* tertinggi sebesar 0,135 dibandingkan dengan *clustering* tanpa pengisian data kosong yang hanya 0,101.
3. Proses *clustering* dengan parameter terbaik yang telah didapatkan memberikan hasil *cluster* dengan nilai *Silhouette Coefficient* tertinggi sebesar 0,351, jumlah sebanyak 2 dan proporsi *cluster* 1:37. *Cluster* dengan id (0,0) hanya beranggotakan kabupaten Jember. Sedangkan, sisa objek data yang lain mengumpul ke *cluster* dengan id (0,1). *Cluster* dengan id (0,0) ini merupakan *outlier*. Lima atribut teratas yang memiliki selisih tertinggi antara cluster id (0,1) dan id (0,0) adalah Lanjut Usia Terlantar, Gelandangan dan Gelandangan Psikotik, Pemulung, Pengemis dan Kelompok Minoritas. Nilai 0,351 termasuk ke dalam kategori *clustering* dengan *weak structure.*

7.2 Saran

Penelitian ini masih terdapat beberapa kekurangan. Pada bagian ini saran ini, harapannya penelitian selanjutnya dapat mengatasi kekurangan-kekurangan yang ada pada penelitian yang telah dilakukan. Saran untuk penelitian selanjutnya adalah sebagai berikut:

1. Penelitian selanjutnya difokuskan untuk melakukan *clustering* dengan seleksi atribut atau fitur. Seleksi fitur yang dilakukan dapat berupa

pemilihan atribut PMKS sesuai dengan konteks penelitian yang akan dilakukan ataupun dapat dengan penggunaan algoritme khusus yang digunakan untuk menyeleksi fitur atau variabel data yang tidak berpengaruh signifikan. Salah satu algoritme yang dapat digunakan untuk seleksi fitur yaitu *Principal Component Analysis* (Xu *et al.*, 2008).

2. Pada penelitian ini, proses analisis *cluster* dan interpretasi *cluster* yang dilakukan masih menggunakan metode manual yaitu melihat rata-rata atribut setiap *cluster* dan melihat selisih atribut dari *cluster-cluster* yang terbentuk. Salah satu metode yang dapat digunakan untuk melakukan interpretasi *cluster* secara otomatis berdasar ciri yang ada di atribut *datanya* yaitu dapat menggunakan *Decision Tree* (Parisot *et al.*, 2014). Pada penelitian tersebut, *rule* pada *Decision Tree* yang digunakan untuk mengintepretasikan label *cluster* yang telah dibuat menggunakan proses *clustering*.
3. Pada penelitian ini, proses *clustering* yang terjadi dengan algoritme *Self-Organizing Maps* (SOM) dapat menghasilkan bentuk *cluster* yang berbeda-beda dengan nilai *Silhouette Coefficient* yang berbeda pula. Hal ini terjadi karena inisialisasi bobot jaringan SOM yang masih *random* sehingga proses training dapat terjebak ke *local optima*. Penggunaan teknik optimasi sejenis *Genetic Algorithm* (GA) dapat digunakan untuk mengoptimasi penentuan bobot jaringan awal yang dipakai (Hassan *et al.*, 2015).
4. Data yang digunakan juga harus disesuaikan dengan kondisi terkini (*up-to-date*. Sehingga proses pengelompokan data dapat merepresentasikan kondisi terkini dari data PMKS itu sendiri. Selain itu, data terkini diperlukan agar pembuatan kebijakan yang berdasarkan pada pengelompokan data PMKS ini tepat sasaran.

DAFTAR REFERENSI

Dehuri, S., Mohapatra, C., Ghosh, A. & Mall, R. 2006. A Comparative Study of Clustering Algorithms. *Information Technology Journal*, 5(3): 551–559. Tersedia di http://www.scialert.net/abstract/?doi=itj.2006.551.559.

Departemen Sosial RI 2012. *Kementerian Sosial Dalam Angka - Pembangunan Kesejahteraan Sosial*. Jakarta: Badan Pendidikan dan Penelitian Kesejahteraan Sosial, Pusat Data dan Informasi Kesejahteraan Sosial.

Faiq, N. & Parmin 2018. *Pemprov Jatim Alokasikan Anggaran Pendidikan 30, 40 Persen*. Surya.co.id. Tersedia di http://surabaya.tribunnews.com/2018/08/15/pemprov-jatim-alokasikan-anggaran-pendidikan-3040-persen [Diakses 15 Januari 2019].

Fausett, L. V. 1993. *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*. 1 ed. Prentice-Hall.

Firdauz, E.H. & Widodo, E. 2018. Pengelompokan Data PMKS Menggunakan Metode Self Organizing Maps. *Jurnal LP3M*, 4(1): 37–44.

Gelman, A. & Hill, J. 2007. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. New York: Cambridge University Press.

Gharib, T.F., Fouad, M.M., Mashat, A. & Bidawi, I. 2012. Self Organizing Map - based Document Clustering Using WordNet Ontologies. *International Journal of Computer Science*, 9(1): 88–95.

Gorunescu, F. 2011. *Data Mining: Concepts, Models and Techniques*. Berlin Heidelberg: Springer.

Hafiludien, A. & Istiawan, D. 2018. Penerapan Algoritma Self Organizing Maps Untuk Pemetaan Penyandang Kesejahteraan Sosial (PMKS) di Provinsi Jawa Tengah Tahun 2016. *The 7th University Research Colloqium*, 84–92.

Hagan, M.T., Demuth, H.B., Beale, M.H. & Jesús, O. De 2014. *Neural Network Design*. 2 ed. Stillwater, Oklahoma: Oklahoma State University.

Handoyo, R., Rumani, R. & Nasution, S.M. 2014. Perbandingan Metode Clustering Menggunakan Metode Single Linkage Dan K-Means Pada Pengelompokan Dokumen. *JSM STMIK Mikroskil*, 15(2): 73–82.

Hassan, A., Andi Purnomo, M.R. & Annisa, P.D. 2015. Clustering Using Genetic Algorithm-Based Self-Organising Map. *Advanced Materials Research*, 1115: 573–577.

Hudin, M.S., Ali, M.F. & Adinugroho, S. 2018. Implementasi Metode Text Mining dan K-Means Clustering untuk Pengelompokan Dokumen Skripsi (Studi Kasus : Universitas Brawijaya). *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer (J-PTIIK) Universitas Brawijaya*, 2(11): 5518–5524.

Jerez, J.M., Molina, I., García-Laencina, P.J., Alba, E., Ribelles, N., Martín, M. & Franco, L. 2010. Missing Data Imputation using Statistical and Machine Learning Methods in A Real Breast Cancer Problem. *Artificial Intelligence in Medicine*, 50(2): 105–115.

Kaufmann, L. & Rousseeuw, P.J. 1990. *FInding Groups in Data*. New York: John Wiley & Sons.

Kohonen, T. 1990. The Self-Organizing Map. *Proceedings of The IEEE*, 78(9): 1464–1480.

Natita, W., Wiboonsak, W. & Dusadee, S. 2016. Appropriate Learning Rate and Neighborhood Function of Self-organizing Map (SOM) for Specific Humidity Pattern Classification over Southern Thailand. *International Journal of Modeling and Optimization*, 6(1): 61–65. Tersedia di http://www.ijmo.org/index.php?m=content&c=index&a=show&catid=61&id=606.

Parisot, O., Ghoniem, M. & Otjacques, B. 2014. Decision Trees and Data Preprocessing to Help Clustering Interpretation. (August): 48–55.

Permensos 2012. *Pedoman Pendataan dan Pengelolaan Data Penyandang Masalah Kesejahteraan Sosial dan Potensi dan Sumber Kesejahteraan Sosial*. Jakarta: Kementrian Sosial (Kemensos). Tersedia di http://peraturan.go.id/inc/view/11e6c5bb4146bec08414313431373532.html.

Pradnyana, G.A. & Permana, A.A.J. 2018. Sistem Pembagian Kelas Kuliah Mahasiswa Dengan Metode K-Means Dan K-Nearest Neighbors Untuk Meningkatkan Kualitas Pembelajaran. *JUTI: Jurnal Ilmiah Teknologi Informasi*, 16(1): 59.

Rahmawati, L., Cahyani, A.D. & Putro, S.S. 2015. Pemanfaatan Metode Cluster SOM – IDB sebagai Analisa Pengelompokan Penerima Beasiswa. *Jurnal Sistem Informasi Indonesia*, 1(1): 11–17.

Rousseeuw, P.J. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20(4): 53–65.

Xu, J.L., Xu, B.W., Zhang, W.F. & Cui, Z.F. 2008. Principal component analysis based feature selection for clustering. *Proceedings of the 7th International Conference on Machine Learning and Cybernetics, ICMLC*, 1(July): 460–465.

LAMPIRAN A DATASET YANG DIGUNAKAN

Tabel 7.1 Dataset PMKS

KAB/KOTA	ANAK JALANAN	TUNA SUSILA	PENGEMIS	GELANDANGAN DAN GELANDANGAN PSIKOTIK	ANAK BALITA TERLANTAR	ANAK TERLANTAR	ANAK BERHADAPAN DENGAN HUKUM	ANAK DENGAN KEDISABILITASAN	ANAK YANG MENJADI KORBAN TEINDAK KEKERASAN ATAU YANG DIPERLAKUKAN SALAH	ANAK YANG MEMERLUKAN PERLINDUNGAN KHUSUS
Pacitan	(kosong)	(kosong)	6	32	(kosong)	(kosong)	4	992	2	(kosong)
Ponorogo	52	(kosong)	153	114	562	983	27	1392	40	3
Trenggalek	16	(kosong)	68	42	4471	19633	20	673	38	111
Tulungagung	6	162	32	13	87	323	23	0	2	0
Blitar	46	(kosong)	88	75	94	188	74	1772	14	41
Kediri	184	107	154	132	8	406	68	1829	29	52
Malang	210	(kosong)	220	139	57	4788	6	453	27	18
Lumajang	15	116	97	6	21	147	20	632	21	13
Jember	171	172	332	265	142	3062	47	2590	31	46
Banyuwangi	(kosong)	(kosong)	297	267	333	1108	4	3001	13	42

KAB/KOTA	ANAK JALANAN	TUNA SUSILA	PENGEMIS	GELANDANGAN DAN GELANDANGAN PSIKOTIK	ANAK BALITA TERLANTAR	ANAK TERLANTAR	ANAK BERHADAPAN DENGAN HUKUM	ANAK DENGAN KEDISABILITASAN	ANAK YANG MENJADI KORBAN TEINDAK KEKERASAN ATAU YANG DIPERLAKUKAN SALAH	ANAK YANG MEMERLUKAN PERLINDUNGAN KHUSUS
Bondowoso	84	50	100	212	100	93	6	136	2	38
Situbondo	10	20	20	(kosong)	0	20	0	70	0	0
Probolinggo	431	136	89	76	593	1300	0	3270	14	0
Pasuruan	120	18	43	58	58	165	23	848	7	42
Sidoarjo	145	20	125	102	81	243	100	0	0	0
Mojokerto	102	99	108	62	1031	5301	15	1028	33	47
Jombang	16	71	71	179	40	94	51	385	37	12
Nganjuk	91	(kosong)	84	97	48	3110	95	1857	26	0
Madiun	93	122	152	14	349	2780	0	578	26	0
Magetan	38	90	7	30	0	5497	26	900	16	23
Ngawi	86	12	106	72	301	10807	4	481	4	0
Bojonegoro	36	61	153	25	223	633	9	1515	16	51
Tuban	31	62	194	196	280	685	65	867	35	53

KAB/KOTA	ANAK JALANAN	TUNA SUSILA	PENGEMIS	GELANDANGAN DAN GELANDANGAN PSIKOTIK	ANAK BALITA TERLANTAR	ANAK TERLANTAR	ANAK BERHADAPAN DENGAN HUKUM	ANAK DENGAN KEDISABILITASAN	ANAK YANG MENJADI KORBAN TEINDAK KEKERASAN ATAU YANG DIPERLAKUKAN SALAH	ANAK YANG MEMERLUKAN PERLINDUNGAN KHUSUS
Lamongan	26	11	10	3	0	3108	4	232	12	0
Gresik	112	122	232	53	4830	46360	42	498	13	18
Bangkalan	12	5	30	12	55	111	1	326	0	0
Sampang	36	1	14	11	64	6920	2	616	1	9
Pameksan	(kosong)	(kosong)	174	7	701	6988	0	1455	28	0
Sumenep	19	(kosong)	114	20	6	498	0	272	0	0
Kota Kediri	18	9	8	16	7	59	4	153	11	8
Kota Blitar	15	71	4	4	25	237	4	90	2	0
Kota Malang	104	4	71	10	5	(kosong)	25	132	1	5
Kota Probolinggo	(kosong)	(kosong)	95	(kosong)	2	384	10	202	24	0
Kota Pasuruan	18	(kosong)	74	1	5	28	4	28	0	14
Kota Mojokerto	6	(kosong)	58	3	8	120	0	50	0	3

KAB/KOTA	ANAK JALANAN	TUNA SUSILA	PENGEMIS	GELANDANGAN DAN GELANDANGAN PSIKOTIK	ANAK BALITA TERLANTAR	ANAK TERLANTAR	ANAK BERHADAPAN DENGAN HUKUM	ANAK DENGAN KEDISABILITASAN	ANAK YANG MENJADI KORBAN TEINDAK KEKERASAN ATAU YANG DIPERLAKUKAN SALAH	ANAK YANG MEMERLUKAN PERLINDUNGAN KHUSUS
Kota Madiun	3	4	33	6	89	282	2	157	1	8
Kota Surabaya	50	5	62	17	19	1059	8	810	0	56
Kota Batu	3	4	5	1	7	47	3	232	2	1

Tabel 7.1 Dataset PMKS (lanjutan)

KAB/KOTA	LANJUT USIA TERLANTAR	PENYANDANG DISABILITAS & BEKAS PENDERITA PENYAKIT KRONIS	PEMULUNG	KELOMPOK MINORITAS	BEKAS WARGA BINAAN LEMBAGA PEMASYARAKATAN	ORANG DENGAN HIV	ORANG DENGAN AIDS	KORBAN PENYALAGUNAAN NAPZA	KORBAN TRAFFICKING	KORBAN TINDAK KEKERASAN ATAU YANG DIPERLAKUKAN SALAH
Pacitan	1499	4150	130	(kosong)	115	39	79	(kosong)	(kosong)	(kosong)
Ponorogo	7916	5033	267	(kosong)	163	92	384	27	133	108
Trenggalek	3868	4798	27	12	187	36	108	352	(kosong)	46
Tulungagung	1918	823	(kosong)	(kosong)	165	441	747	37	(kosong)	51

KAB/KOTA	LANJUT USIA TERLANTAR	PENYANDANG DISABILITAS & BEKAS PENDERITA PENYAKIT KRONIS	PEMULUNG	KELOMPOK MINORITAS	BEKAS WARGA BINAAN LEMBAGA PEMASYARAKATAN	ORANG DENGAN HIV	ORANG DENGAN AIDS	KORBAN PENYALAGUNAAN NAPZA	KORBAN TRAFFICKING	KORBAN TINDAK KEKERASAN ATAU YANG DIPERLAKUKAN SALAH
Blitar	619	4539	229	26	636	83	866	206	11	21
Kediri	2507	3965	490	88	715	222	501	233	9	86
Malang	4541	22041	(kosong)	(kosong)	1418	234	1414	135	(kosong)	5
Lumajang	1811	3848	124	(kosong)	314	164	200	110	2	30
Jember	16288	4185	776	489	709	639	890	555	21	120
Banyuwangi	9393	3493	(kosong)	(kosong)	84	361	962	288	(kosong)	37
Bondowoso	3810	1866	(kosong)	502	50	7	70	40	(kosong)	20
Situbondo	33	70	(kosong)	(kosong)	10	199	99	37	(kosong)	(kosong)
Probolinggo	5446	3141	(kosong)	(kosong)	615	229	591	118	(kosong)	(kosong)
Pasuruan	1803	1363	288	240	41	244	1476	7	2	47
Sidoarjo	1292	2585	(kosong)	(kosong)	40	437	1041	38	(kosong)	(kosong)
Mojokerto	7761	3935	248	130	373	49	185	47	5	16
Jombang	2983	2373	127	47	130	161	283	36	2	86

KAB/KOTA	LANJUT USIA TERLANTAR	PENYANDANG DISABILITAS & BEKAS PENDERITA PENYAKIT KRONIS	PEMULUNG	KELOMPOK MINORITAS	BEKAS WARGA BINAAN LEMBAGA PEMASYARAKATAN	ORANG DENGAN HIV	ORANG DENGAN AIDS	KORBAN PENYALAGUNAAN NAPZA	KORBAN TRAFFICKING	KORBAN TINDAK KEKERASAN ATAU YANG DIPERLAKUKAN SALAH
Nganjuk	4058	2705	(kosong)	27	255	183	547	268	(kosong)	91
Madiun	2968	2637	(kosong)	(kosong)	243	24	266	3	(kosong)	90
Magetan	5695	6286	109	(kosong)	249	43	150	3	3	52
Ngawi	14352	3626	215	(kosong)	2400	92	224	(kosong)	(kosong)	1
Bojonegoro	4904	5505	436	54	211	163	367	10	(kosong)	47
Tuban	7374	4421	645	7	398	78	581	81	3	25
Lamongan	3188	899	24	(kosong)	(kosong)	164	710	0	(kosong)	(kosong)
Gresik	5283	3017	160	241	77	164	898	53	(kosong)	148
Bangkalan	472	1431	37	(kosong)	35	55	74	(kosong)	(kosong)	15
Sampang	3005	2806	38	(kosong)	318	2	112	3	(kosong)	(kosong)
Pameksan	2992	6208	15	(kosong)	(kosong)	6	16	3	(kosong)	(kosong)
Sumenep	3267	1760	425	(kosong)	66	25	87	6	(kosong)	148
Kota Kediri	423	200	12	(kosong)	41	214	111	558	(kosong)	18

KAB/KOTA	LANJUT USIA TERLANTAR	PENYANDANG DISABILITAS & BEKAS PENDERITA PENYAKIT KRONIS	PEMULUNG	KELOMPOK MINORITAS	BEKAS WARGA BINAAN LEMBAGA PEMASYARAKATAN	ORANG DENGAN HIV	ORANG DENGAN AIDS	KORBAN PENYALAGUNAAN NAPZA	KORBAN TRAFFICKING	KORBAN TINDAK KEKERASAN ATAU YANG DIPERLAKUKAN SALAH
Kota Blitar	342	354	11	(kosong)	119	57	33	6	(kosong)	(kosong)
Kota Malang	1179	1397	120	39	44	311	684	878	1	4
Kota Probolinggo	231	1007	(kosong)	(kosong)	14	47	45	(kosong)	(kosong)	29
Kota Pasuruan	219	206	38	(kosong)	100	31	153	1	(kosong)	1
Kota Mojokerto	1344	375	33	(kosong)	(kosong)	135	41	(kosong)	(kosong)	2
Kota Madiun	771	508	64	(kosong)	51	125	106	15	(kosong)	(kosong)
Kota Surabaya	6912	6514	175	(kosong)	44	953	2170	31	(kosong)	(kosong)
Kota Batu	339	552	57	(kosong)	20	45	147	6	(kosong)	2

Tabel 7.1 Dataset PMKS (lanjutan)

KAB/KOTA	PEKERJA MIGRAN BERMASALAH SOSIAL	KORBAN BENCAN ALAM	KORBAN BENCANA SOSIAL / PENGUNGSI	PEREMPUAN RAWAN SOSIAL EKONOMI	KELUARGA BERMASALAH SOSIAL PSIKOLOGIS	MASYARAKAT DAERAH TERTINGGAL DAN TERPENCIL	KELUARGA FAKIR MISKIN
Pacitan	490	99	(kosong)	1859	104	139	(kosong)
Ponorogo	432	440	724	11	3	267	(kosong)
Trenggalek	20	308	(kosong)	8512	102	1268	(kosong)
Tulungagung	4	550	1	2260	60	(kosong)	(kosong)
Blitar	57	595	22	1397	237	(kosong)	(kosong)
Kediri	62	167	13	10686	750	(kosong)	(kosong)
Malang	50	285	21	12438	907	(kosong)	(kosong)
Lumajang	131	130	23	909	182	(kosong)	(kosong)
Jember	101	382	44	6012	(kosong)	(kosong)	(kosong)
Banyuwangi	0	590	(kosong)	5684	298	1310	(kosong)
Bondowoso	20	(kosong)	(kosong)	(kosong)	(kosong)	(kosong)	(kosong)
Situbondo	0	(kosong)	(kosong)	(kosong)	(kosong)	40	(kosong)
Probolinggo	373	272	(kosong)	4441	235	(kosong)	(kosong)
Pasuruan	0	26	55	1450	191	(kosong)	(kosong)

KAB/KOTA	PEKERJA MIGRAN BERMASALAH SOSIAL	KORBAN BENCAN ALAM	KORBAN BENCANA SOSIAL / PENGUNGSI	PEREMPUAN RAWAN SOSIAL EKONOMI	KELUARGA BERMASALAH SOSIAL PSIKOLOGIS	MASYARAKAT DAERAH TERTINGGAL DAN TERPENCIL	KELUARGA FAKIR MISKIN
Sidoarjo	0	412	47	2325	66	(kosong)	(kosong)
Mojokerto	26	426	(kosong)	4888	181	(kosong)	(kosong)
Jombang	40	(kosong)	4	501	84	(kosong)	(kosong)
Nganjuk	0	120	78	5600	381	(kosong)	(kosong)
Madiun	52	570	(kosong)	3253	100	(kosong)	(kosong)
Magetan	0	600	(kosong)	474	(kosong)	(kosong)	(kosong)
Ngawi	0	1461	6	5854	(kosong)	(kosong)	(kosong)
Bojonegoro	8	1080	18	4858	190	16	(kosong)
Tuban	25	173	2	5026	75	(kosong)	(kosong)
Lamongan	30	35340	29	24	(kosong)	(kosong)	(kosong)
Gresik	215	4350	68	4734	56	(kosong)	(kosong)
Bangkalan	47	781	44	2141	24	(kosong)	(kosong)
Sampang	350	249	93	5895	(kosong)	(kosong)	(kosong)
Pameksan	584	559	7	3	1570	1	(kosong)
Sumenep	267	138	(kosong)	3153	14	(kosong)	(kosong)

KAB/KOTA	PEKERJA MIGRAN BERMASALAH SOSIAL	KORBAN BENCAN ALAM	KORBAN BENCANA SOSIAL / PENGUNGSI	PEREMPUAN RAWAN SOSIAL EKONOMI	KELUARGA BERMASALAH SOSIAL PSIKOLOGIS	MASYARAKAT DAERAH TERTINGGAL DAN TERPENCIL	KELUARGA FAKIR MISKIN
Kota Kediri	0	10	(kosong)	334	28	(kosong)	(kosong)
Kota Blitar	1	4	1	820	205	(kosong)	(kosong)
Kota Malang	0	2	196	919	(kosong)	(kosong)	(kosong)
Kota Probolinggo	0	(kosong)	(kosong)	5616	(kosong)	(kosong)	(kosong)
Kota Pasuruan	0	(kosong)	(kosong)	2555	14	(kosong)	(kosong)
Kota Mojokerto	0	(kosong)	(kosong)	965	4	(kosong)	(kosong)
Kota Madiun	0	5	(kosong)	829	29	(kosong)	(kosong)
Kota Surabaya	0	26	1	2168	78	(kosong)	(kosong)
Kota Batu	1	(kosong)	4	192	3	(kosong)	(kosong)